Institutional Repository - Research Portal
Dépôt Institutionnel - Portail de la Recherche
researchportal.unamur.be
University of Namur

THESIS / THÈSE

MASTER EN SCIENCES INFORMATIQUES

La table de décision séquentielle
Outil de synthèse des procédures séquentielles

Mannes, Colette

Award date:
1975

Awarding institution:
Universite de Namur

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal ?

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 05. May. 2023

FACULTES UNIVERSITAIRES NOTRE-DAME DE LA PAIX - NAMUR

INSTITUT D'INFORMATIQUE

Année académique 1974-1975

Institut d'informatique
Bibliothèque
Tél. 081-747.49 FNDP NAMUR

LA TABLE DE DÉCISION SÉQUENTIELLE

OUTIL DE SYNTHESE DES PROCEDURES SEQUENTIELLES

Colette MANNES

MEMOIRE PRESENTE EN VUE DE L'OBTENTION DU GRADE DE LICENCIE ET MAITRE EN INFORMATIQUE

Jury du mémoire : M.J. BRUNIN

A mes parents,

Nous tenons à exprimer notre profonde gratitude à Monsieur le Professeur J. BRUNIN, pour les conseils bienveillants qu'il nous a donnés tout au long de ce mémoire et pour le profit que nous avons pu tirer de son expérience.
Qu'il trouve ici l'expression de toute notre reconnaissance pour nous avoir permis de réaliser et de mener à bien notre tâche.

Nous remercions vivement Monsieur le Professeur B. FILLIATRE pour l'accueil qu'il nous a réservé à Montpellier et pour les critiques constructives qu'il nous a faites.

Nous remercions également Monsieur le Professeur E. MORLET pour l'attention qu'il nous a prêtée et les suggestions qu'il nous a faites.

Enfin, nous n'oublions pas tous ceux qui par leur patience et leur soutien nous ont aidé anonymement.

TABLE DES MATIERES

INTRODUCTION

Notre travail a pour objet la synthèse des procédures séquentielles en logique enregistrée.

Une procédure séquentielle est définie comme une suite d'actions qui se déroulent impérativement dans un ordre donné et qui sont reliées entre elles par des articulations appelées conditions.

La synthèse de toute procédure en logique enregistrée, revêt, dans l'ordre, les quatre points suivants :

- la mise en page,
- l'analyse,
- la simplification,
- la préparation à la programmation.

La synthèse des procédures séquentielles est d'habitude conçue et réalisée au moyen d'un organigramme. Cependant, il serait souvent agréable de disposer d'un outil plus manipulable, plus concis et plus performant du point de vue synthèse.

L'outil tabulaire est pratique; nous en recontrons beaucoup tout autour de nous. Il est un moyen d'ordonner des informations. Il peut donc aussi être un moyen de description pour présenter l'enchaînement parfois complexe d'actions soumises à des conditions.

L'outil de synthèse dont nous allons aborder l'étude est un outil tabulaire : la table de décision séquentielle.

Au cours d'une première partie, nous parlerons de l'origine de la table de décision séquentielle et de sa structure. Nous l'examinerons en tant qu'outil de synthèse. Dans une seconde partie, nous l'utiliserons pour synthétiser des problèmes concrets.

CHAPITRE I

POURQUOI UN NOUVEL OUTIL DE SYNTHESE ?

1.1. Procédure combinatoire et procédure séquentielle.

Une procédure qu'elle soit à implémenter en logique câblée ou en logique enregistrée, peut être de nature :

- soit combinatoire
- soit séquentielle.

Une procédure est de nature combinatoire si son résultat n'est fonction ni d'un ordre d'examen des conditions, ni d'un ordre d'exécution des décisions; elle est de nature séquentielle, si au contraire, un ordre affectant les conditions à examiner, ou les décisions à exécuter, ou les deux, doit être pris en considération pour aboutir au résultat désiré.

Il convient cependant de remarquer qu'une procédure de nature combinatoire à implémenter en logique enregistrée va perdre sa nature combinatoire pour devenir séquentielle. Elle est alors séquentielle par nécessité, pour des raisons technologiques et d'implémentation, non par nature.

Donnons un exemple de procédure de chaque type.

Le problème suivant de la réservation d'une place d'avion est un problème combinatoire : "Si le client demande une place touriste (DT) ou première (Dp) et que le vol est complet, la place lui est refusée (R). Si le vol n'est pas complet, que le client demande une place touriste et qu'il y en a encore de libre (dt) une place touriste (T) lui est réservée; sinon s'il n'y a plus de place disponible en classe touriste, la compagnie propose au client une place de première que celui-ci accepte ou refuse selon qu'il est d'accord ou non (A) de changer de classe. Si le vol n'est pas complet, que le client désire une place première et qu'il y en a de libre (dp), une place lui est réservée (P) sinon la compagnie propose au client une place touriste qu'il peut accepter ou refuser".

La nature de ce problème est combinatoire ou encore logique à savoir qu'il peut être mis sous forme d'équations logiques (utilisation des opérateurs ET et OU).

Un exemple de procédure séquentielle est le suivant : "lors de la mise à jour d'un fichier, chaque article est susceptible de subir un certain nombre d'opérations :

- création
- annulation
- modification.

Il s'agit pour chaque article d'analyser la séquence d'opérations qui l'affecte, afin de déterminer si elle est correcte ou non sachant que :

- une création ne peut suivre ni une opération de modification ni une opération de création;
- une annulation ne peut succéder à une annulation;
- une modification ne peut succéder à une annulation.

Nous constatons que le résultat est fonction d'un ordre affectant les conditions : ainsi si se présente une demande d'annulation suivie d'une demande de modification, on conclura à une erreur, par contre si se présente une demande de modification suivie d'une demande d'annulation on conclura à une annulation.

1. 2. Tables de décision et synthèse des procédures combinatoires.

L'outil universel de synthèse des procédures à implémenter en logique enregistrée est l'organigramme. Il était le seul connu et enseigné aux programmeurs et analystes jusqu'il y a peu.

Il y a une quinzaine d'années, a été développé un nouvel outil de synthèse des procédures combinatoires : la table de décision.

La table de décision (que nous appellerons combinatoires pour la distinguer de la table de décision séquentielle que nous verrons plus loin) est l'outil de synthèse par excellence des procédures combinatoires : en effet, elle n'est pas de nature séquentielle et par là plus proche du problème à traiter que ne l'est l'organigramme, qui est lui séquentiel par essence et déjà orienté vers l'ordinateur.

Elle décompose la procédure de façon à en faire un tableau clair, lisible par tous, assurant une analyse aisée et éventuellement point de départ d'une programmation automatique ("DLT" : decision logic translator d'IBM qui convertit les tables en un programme fortran).

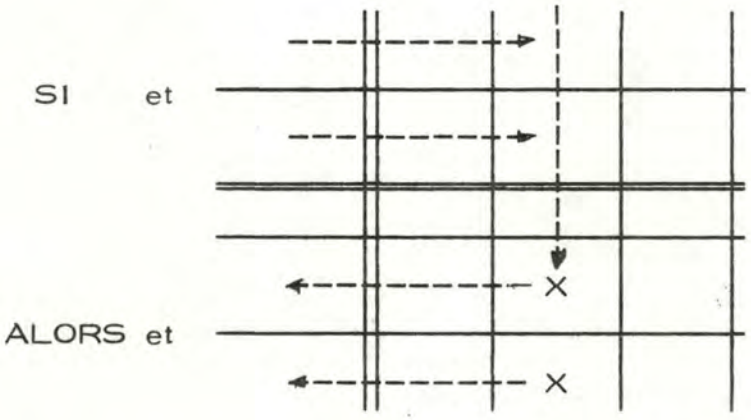
1. 2. 1. Rappel de la structure et des types de table de décision.

Rappelons brièvement la structure des tables de décision: la présentation générale de la table est toujours la suivante :

Souche "Conditions"	Entrées Condition
Souche "Actions"	Entrées Action

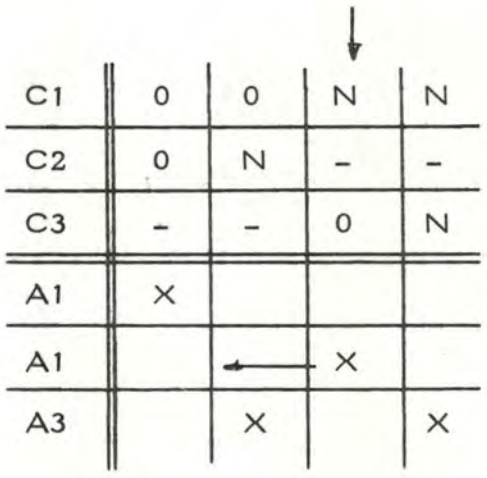
à gauche de la table apparait ce que l'on appelle la souche et à droite les entrées qui comportent un certain nombre de colonnes. Nous distinguons en haut, la souche et les entrées des conditions et en bas, la souche et les entrées des actions. La souche des conditions énumère les conditions du problème traité dans la table; la souche des actions, toutes les actions à exécuter dans le cadre du problème. L'entrée des conditions permet d'indiquer si les conditions énumérées sont remplies ou non. Une condition peut aussi être indifférente c'est-à-dire que vérifée ou non, elle n'a aucune influence sur les actions à exécuter. L'entrée des actions indique d'effectuer les actions commandées par l'ensemble des conditions remplies.

On matérialise l'entrée des actions en mettant une "X" ou un opérande en face des actions à effectuer.



Chaque colonne des entrées est, en fait, une relation du type : Si certaines conditions sont remplies, ALORS certaines actions sont à exécuter. On appelle cette relation une règle de décision. L'ensemble de ces règles de décision doit reçouvrir l'ensemble des cas possibles sans omission, ni répétition.

Exemple :



C1	0	0	N	N
C2	0	N	-	-
C3	-	-	0	N
A1	X			
A1		←	X	
A3		X		X

SI la condition C1 n'est pas remplie ET SI la condition C2 est indifférente, ET SI la condition C3 est remplie ALORS effectuer l'action A2.

Le fonctionnement d'une table lors d'un passage est le suivant :

- un seul ensemble des conditions est rempli;
- une seule règle de décision est retenue;
- une seule colonne est utilisée;
- et donc un seul ensemble d'actions est réalisé.

A chaque passage, on ne peut prendre qu'une seule décision par table.

Il existe 3 types de table combinatoire :

1. Table à entrée limitée : elle présente les caractéristiques suivantes :

   - conditions et actions sont entièrement écrites dans la souche;
   - l'entrée des conditions ne peut être que oui (O) ou non (N);
   - l'entrée des actions ne peut être que "X" si l'action est effectuée ou "blanc " dans le cas contraire.

   Exemple : l'accord d'un crédit.

		R1	R2	R3
C1	limite de crédit OK	0	N	N
C2	expérience de paie favorable	-	O	N
A1	approuver le crédit	X	X	-
A2	envoyer un ordre de paiement	-	-	X

2. Table à entrée étendue :

   - l'entrée des conditions et des actions contient une partie de la condition ou de l'action;
   - la souche contient seulement un des termes de la comparaison, l'autre se trouvant dans les colonnes.

C1	limite de crédit	OK	non OK	non OK
C2	expérience de paie	-	favorable	défavorable
A1	crédit	approuvé	approuvé	non approuvé
A2	ordre à envoyer	-	-	paiement

3. Table à entrée mixte : elles contiennent à la fois des lignes à entrée limitée et des lignes à entrée étendue.

C1	limite de crédit OK	O	N	N
C2	expérience de paie	-	favorable	défavorable
A1	crédit	approuvé	approuvé	non approuvé
A2	envoyer ordre de paiement	-	-	X

1. 2. 2. Exemple.

Nous ne nous attarderons pas d'avantage sur la synthèse des procédures combinatoires au moyen des tables de décision; celle-ci a déjà fait l'objet de nombreux ouvrages et publications.

Donnons simplement pour terminer la mise en page de l'exemple de procédure combinatoire du paragraphe 1. 1.

DP	O	O	O	O	N	N	N	N
DT	N	N	N	N	O	O	O	O
dp	O	N	N	N	-	O	O	N
dt	-	N	O	O	O	N	N	N
A	-	-	O	N	-	O	N	-
R		X		X			X	X
T			X		X			
P	X					X		

1. 3. Tables de décision et synthèse des procédures séquentielles.

Le mémoire a pour objet d'étude les procédures séquentielles dont la succession des opérations n'est plus imposée par la technologie mais par l'application elle-même.

Le problème de la mise en page d'une procédure séquentielle réside donc dans l'enregistrement de l'ordre de succession

des différentes conditions à tester et décisions à prendre ou traitements à exécuter (lorsque nous parlerons de traitement nous entendons par là un ensemble homogène de traitements élémentaires; un ordre peut affecter les traitements élémentaires au sein d'un même ensemble).

Une première question qui vient à l'esprit est la suivante : existe-t-il des procédures de nature séquentielle pouvant être synthétisées au moyen de tables combinatoires ? Ou encore : dans quelle mesure les tables combinatoires constituent-elles un outil de synthèse pour les procédures séquentielles ?

1. 3. 1. Synthèse de procédures séquentielles par table combinatoire.

Bien que la table combinatoire ne soit pas destinée à traduire la nature séquentielle, il existe des cas de procédures qui sont séquentielles et qui peuvent être synthétisées "aisément" au moyen de tables de décision classiques. Parmi eux, notons les cas suivants :

- la nature séquentielle de la procédure est due à un ordre qui n'affecte que certaines conditions ou certaines décisions;
- la procédure se ramène à l'exploitation séquentielle d'un ou de plusieurs modules combinatoires;
- une combinaison des deux cas précédents.

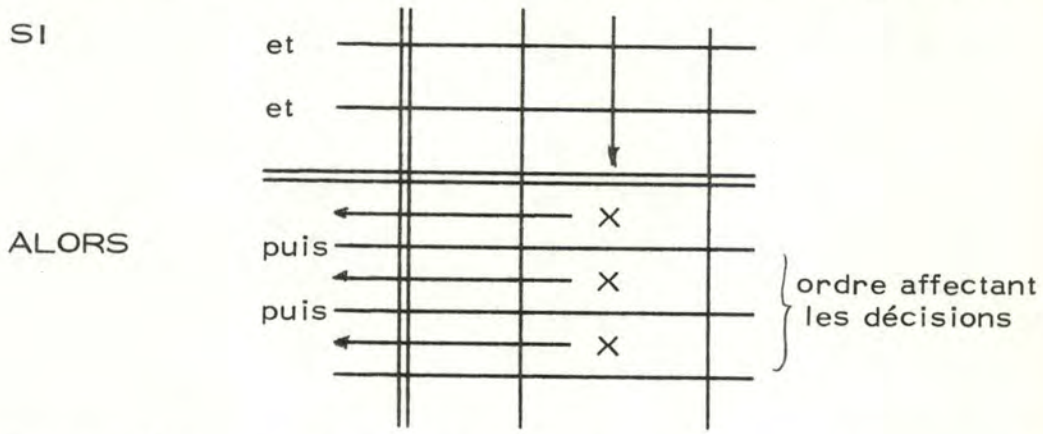
1. 3. 1. 1. Ordre unique affectant certains traitements ou certaines conditions.

Il s'agit de procédures qui doivent leur nature séquentielle à un ordre qui n'affecte que certaines conditions à tester ou que certaines décisions à prendre et ce de manière telle qu'étant donné deux conditions dont dépendent des décisions, elles sont toujours testées dans le même ordre (idem si c'est un ordre qui n'affecte que les décisions).

Dans ce cas, la procédure peut être enregistrée dans une table de décision classique à condition de faire figurer les conditions et les décisions dans leur souche respective, non plus de manière tout à fait quelconque comme dans une table classique mais en respectant l'ordre qui affecte certains d'entre eux.

Exemple : considérons le cas fréquent où les traitements sont liés entre eux en ce sens que certains utilisent des informations fournies par d'autres. Ils doivent donc être exécutés dans un ordre bien déterminé.

La table de décision classique s'interprête alors comme suit :



Le problème suivant est relatif à la facturation des commandes effectuées par la clientèle dans des conditions spécifiques. Voici les directives de calcul.

a. les clients sont répartis en 3 catégories : 1 - 2 -3 suivant le montant de leurs commandes antérieures.
Les ristournes R sont accordées selon ce qui suit :

- un gros client (c = 1) commandant plus de 500 articles se voit accorder une ristourne de 7 %, sinon 4 %;
- un client moyen (c = 2) bénéficie d'une ristourne de 4 %, s'il commande plus de 300 articles, sinon 2 %;
- un petit client (c = 3), s'il commande plus de 150 articles reçoit une ristourne de 3 %.

b. 3 qualités d'articles sont disponibles à 3 prix (hors TVA) différents : P1 - P2 - P3.

La procédure de calcul de la note de paiement est enregistrée dans la table de décision à entrées mixtes :

CONDITIONS	Qualité	1	1	1	1	1	1	2	2	2	2	2	2	3	3	3	3	3	3
	Catégorie	1	1	2	2	3	3	1	1	2	2	3	3	1	1	2	2	3	3
	Quantité	⩽500	>500	⩽300	>300	⩽150	>150	⩽500	>500	⩽300	>300	⩽150	>150	⩽500	>500	⩽300	>300	⩽150	>150
DECISIONS	Prix unitaire	P1	P1	P1	P1	P1	P1	P2	P2	P2	P2	P2	P2	P3	P3	P3	P3	P3	P3
	Ristourne	4%	7%	2%	4%	-	3%	4%	7%	2%	4%	-	3%	4%	7%	2%	4%	-	3%
	Prix hors TVA = quantité * prix unitaire - Ristourne	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	TVA = Prix hors TVA $* \frac{16}{100}$	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	Prix total = prix hors TVA + TVA	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Les traitements ont été enregistrés dans l'ordre : à savoir qu'il faut décider des prix unitaires et de la ristourne avant de pouvoir calculer le prix hors TVA et celui-ci avant de pouvoir calculer la TVA pour enfin déterminer le prix total.

Lorsqu'une telle procédure est enregistrée, nous pouvons procéder à son analyse en suivant les mêmes démarches que pour les procédures combinatoires c'est-à-dire détecter les cas oubliés, incompatibles, redondants, pour ensuite la simplifier et la programmer. Soulignons toutefois qu'il faut programmer les traitements non dans un ordre quelconque mais dans l'ordre dans lequel ils figurent dans la table.

1. 3. 1. 2. Exploitation séquentielle de modules combinatoires.

Certaines procédures séquentielles peuvent se ramener à l'exploitation séquentielle d'un ou de plusieurs modules combinatoires. Ces modules sont chacun synthétisés dans une table de décision combinatoire classique et sont chaînés les uns aux autres. Cette technique de chaînage est la même que celle utilisée pour éclater les tables combinatoires en tables plus petites. Un chaînage peut être soit du type "ouvert" c'est-à-dire représenter un branchement inconditionnel sans retour à la table origine, soit du type "fermé" c'est-à-dire représenter un branchement inconditionnel avec retour à la table origine.

Si la procédure est enregistrée dans un seul module combinatoire, la nature séquentielle apparaît dans le bouclage effectué par le module sur lui-même; s'il y en a plusieurs, la nature séquentielle est traduite par l'enchaînement de ces modules.

Exemple.

" Le salaire mensuel des employés d'une entreprise est calculé de la manière suivante :

- le barème unique de paie est multiplié par un coefficient X ou Y (D1) selon que l'employé est cadre ou ouvrier(C1);
- en fonction de son nombre d'années d'ancienneté, 5, 10, 20 ans (C2); il voit son salaire augmenter de 25, 50 , 75 % (D2);
- s'il est marié - (C3) et que son salaire de base est inférieur à une valeur fixée (C4), l'employé percevra une indemnité de personnes à charge de 2, 5, 10 % (D3) s'il a au moins 2, 4, 6 enfants (C3);
- lorsque le total du salaire de base et de l'indemnité est connu (salaire brut), un test de la valeur de ce cumul par rapport à une valeur fixée déterminera une retenue de 8 % (D4) si la première est supérieure ou égale à la seconde (D6)".

Cette procédure séquentielle est caractérisée par un ordre affectant l'examen des conditions (Ci; i : 1 .... 6) et l'exécution des décisions/traitements (D1, D2, D3, salaire de base, salaire brut, salaire net). Sa nature séquentielle se limite à une exploitation séquentielle de 3 modules combinatoires. Pour s'en rendre compte, il suffit de décomposer l'ensemble des conditions et décisions/traitements en sous-ensembles de manière à ce que en leur attribuant à chacun une table de décision, le déroulement de la séquence corresponde au passage d'une table à l'autre via GOTO (branchement inconditionnel sans retour) ou DO/RETURN (branchement avec retour à la table origine).

Après mise en page, dans ce cas comme dans le précédent, la suite de la synthèse s'opère comme pour la synthèse des procédures combinatoires.

TABLE T1

Conditions	C1 1. cadre 2. ouvrier	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2
	C2 (ancienneté) 1. ≤ 5 2. 5 < - ≤ 10 3. 10 < - ≤ 20 4. 20 <	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
	C3 1. marié 2. Non marié	1	1	1	1	2	2	2	2	1	1	1	1	2	2	2	2
Décisions	D1 (coefficient)	x	x	x	x	x	x	x	x	y	y	y	y	y	y	y	y
	D2 (augmentation %)	-	25	50	75	-	25	50	75	-	25	50	75	-	25	50	75
	Salaire de base	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
	Do T2	x	x	x	x	-	-	-	-	x	x	x	x	-	-	-	-
	Salaire brut (base + indemnité)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
	GOTO T3	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

TABLE T2

Conditions	C4 (salaire base '/. valeur fixée)	<	<	<	<	≥	≥	≥	≥
	C5 (enfants) 1. - de 2 2. au - 2 3. au - 4 4. au - 6	1	2	3	4	1	2	3	4
Déci-sions	D3 (indemnité %)	-	2	5	10	-	-	-	-
	RETURN	x	x	x	x	x	x	x	x

TABLE T3

Cond	C6 (salaire brut '/. valeur fixée)	≥	<
Déci-sions	D4 (retenue de 2 %)	x	-
	Salaire net	x	x

1. 3. 2. Problème de la mémorisation de situations passées.

Les procédures séquentielles jouissent d'une caractéristique qui leur est propre :

Dépendant de la manière dont est perçue et enregistrée une procédure séquentielle par l'analyste, il arrive fréquemment que les contenus des opérations constituant la séquence (test de conditions et prise de décisions ou exécution de traitements), influencent la suite de son déroulement. Autrement dit, il est nécessaire en certains points de la séquence de "se souvenir" de situations passées, de prendre en considération le(s) résultat(s) de sous-séquences antérieures. Ces résultats antérieurs, ces situations passées vont donc faire l'objet de mémorisations. Ces mémorisations doivent, si nécessaire, pouvoir apparaître au sein des tables de décision combinatoires; elles figureront sous forme d'un ou de plusieurs paramètres constituant la trace d'un état précédent.

Exemple :

" Nous avons 2 fichiers sur bande magnétique, par exemple, qu'il s'agit de comparer. Soit A le fichier principal et B le fichier mouvement, et admettons qu'il puisse y avoir plusieurs enregistrements B pour un enregistrement A. Pour simplifier l'exemple, le problème des fins de fichier n'y figure pas, ni celui de l'erreur de séquence. Les traitements à effectuer sont différents selon qu'il y a égalité ou non,

- s'il y a inégalité : l'enregistrement A n'a pas d'enregistrement mouvement correspondant et on le recopie sur un fichier nouveau;
- s'il y a égalité : il faut stocker l'enregistrement mouvement et relire le fichier B pour savoir s'il y a un autre mouvement correspondant au même enregistrement principal A; s'il n'y en a pas, on ne peut plus suivre la même procédure que pour le cas d'inégalité précédemment prévu ; il faut se "souvenir" qu'on a déjà trouvé une correspondance entre les 2 fichiers et traiter le cas d'égalité avant de continuer ".

Cette procédure est enregistrée dans la table classique T.

Table T.

CONDITIONS	Correspondance A et B déjà trouvée ? (TEST = 1 ?)	O	O	N	N
	Fic. A = Fic. B ?	O	N	O	N
	Fic. A ≠ Fic. B ?	N	O	N	O
DECISIONS	Ecrire fichier principal sur fichier nouveau				X
	Ecrire fichier principal modifié sur fichier nouveau		X		
	Ecrire 1 dans TEST			X	
	Ecrire 0 dans TEST		X		
	Stocker enregistrement mouvement	X		x	
	Lire fichier principal (Fic. A)		X		X
	Lire fichier mouvement (Fic. B)	X		X	
	GOTO T	X	X	X	X

La séquence a pu être traduite grâce :

- au paramètre de mémorisation TEST qui constitue la trace d'un état précédent;
- à un ordre qui affecte les traitements dans leur souche : "stocker l'enregistrement mouvement " doit précéder "lire fichier mouvement" ; "écrire fichier principal sur fichier nouveau" doit précéder "lire fichier principal".

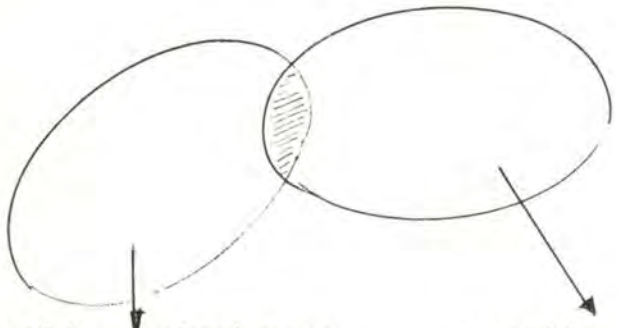
1. 3. 3. Conclusion : procédures simplement séquentielles et procédures fondamentalement séquentielles.

Nous pouvons conclure en disant que les tables de décision classiques peuvent se prêter à la synthèse de procédures séquentielles et ce en utilisant l'une ou l'autre des techniques :

- chaînage de modules combinatoires;
- bouclage sur un même module;
- ordre affectant les conditions ou les traitements/ décisions dans leur souche respective;
- paramètres de mémorisation d'état.

Les procédures séquentielles dont la synthèse par tables de décision combinatoires n'offre aucune difficulté, sont des procédures qui comme nous pouvions nous y attendre et pouvons le constater à l'aide des quelques exemples donnés, sont "plus combinatoires que séquentielles".
La nature séquentielle de ces procédures n'est pas fondamentale au problème; leur nature réelle est combinatoire et sur elle vient de greffer une nature séquentielle secondaire. Elles sont appelées : "procédures simplement séquentielles".

Il est très difficile de définir une limite entre procédures simplement séquentielles et procédures fondamentalement séquentielles c'est-à-dire qu'il existe des procédures dont l'appartenance à l'un ou l'autre type n'est pas décidable.
En prenant les notations de la théorie des ensembles, ceci est traduit par le diagramme de Venn :



procédures simplement séquentielles

Exemples :

- mise à jour de fichier
- grand nombre de problèmes de gestion.

procédures fondamentalement séquentielles

Exemples :

- procédure de gestion des échanges en télétraitement par caractères de contrôle.
- analyse syntaxique d'une expression ou d'une instruction.

La difficulté que nous éprouvons à définir la limite entre les 2 types de procédures séquentielles est due au fait qu'elle est déterminée par des critères qualitatifs :

Si, ayant une procédure séquentielle à synthétiser, l'analyste se rend compte que la synthèse de cette procédure par tables combinatoires, en faisant appel aux techniques précitées, est aisée et que les tables établies sont claires et lisibles (c'est ici que tout est relatif !), c'est que la procédure est plus combinatoire que séquentielle : elle est simplement séquentielle.

Sinon, si la synthèse par tables combinatoires, nécessite des chaînages nombreux et des mémorisations multiples, si la mise en page devient compliquée et illisible, c'est que la nature séquentielle de la procédure est profonde.

1.4. Un nouvel outil de synthèse : la table de décision séquentielle.

Toute procédure séquentielle peut être synthétisée au moyen de tables de décision classiques, en faisant appel au techniques vues plus haut pour traduire la nature séquentielle. Mais cela devient insensé si la séquence est fondamentale; en effet, dans ce cas il faut faire appel à des chaînages de modules multiples et touffus, des mémorisations d'états en nombre important et d'une certaine durée, une découpe de la procédure en modules composés à la limite d'une seule condition à tester, ce qui finalement nous ramène à la structure d'organigramme.

Exemple :

" L'analyse d'une suite de caractères a et b doit déterminer trois types de traitement.
Le traitement T1 s'exécutera lors de l'apparition d'un caractère b suivant un ou une suite de caractères a pour autant que ce ou ces derniers aient été précédés d'un b ou d'une suite de b.
Le traitement T2 traite les caractères b et le traitement T3 les caractères a rencontrés ".
Essayons de mettre en page cette procédure très élémentaire, par tables combinatoires : nous obtenons :

Table 1

caractère	a	b
T2		×
T3	×	
GOTO table 1	×	
GOTO table 2		×

Table 2

Caractère	a	b
T2		x
T3	x	
GOTO table 2		x
GOTO table 3		

Table 3

Caractère	a	b
T1		x
T2		x
T3	x	
GOTO table 3	x	
GOTO table 2		x

Remarquons que dans chaque table, une seule conditions est testée et que ceci n'est rien d'autre qu'un organigramme déguisé !

On peut aussi penser à une procédure caractérisée par un ordre affectant les conditions et les traitements et où chaque condition à tester est positionnée par le(s) traitement(s) qui la précède.

Dans ces cas, les tables de décision combinatoires ne répondent plus à leur objectif et raison d'être "être claires et lisibles". Elles sont donc limitées dans leurs possibilités; elles permettent d'aborder les problèmes de nature combinatoire et tout au plus les problèmes simplement séquentiels. Dans les autres cas, elles parviennent très mal à rendre la nature séquentielle; elles deviennent vite lourdes et confuses (tant à élaborer qu'à utiliser et lire ensuite).

Pour mettre en page et analyser un problème séquentiel fondamentalement, il faut s'en tenir strictement à cette nature.

L'outil de synthèse qui vient à l'esprit est l'organigramme. Il permet la mise en page de toute procédure à implémenter en logique enregistrée (donc séquentielle si non par nature, du moins par nécessité) et est le document de travail du programmeur. Mais est-il un bon outil d'analyse et de simplification ? Permet-il de vérifier si tous les cas ont été envisagés et enregistrés ou si la procédure ne risque pas de boucler lors de son déroulement, sinon en faisant appel aux théories développées pour démontrer la fiabilité des programmes ?

D'autre part, il serait aussi souvent agréable de disposer d'un outil plus concis et plus manipulable que l'organigramme.

C'est pourquoi, partant de deux idées qui sont :

- chercher un outil de synthèse (dans la pleine signification du terme) pour les procédures séquentielles au même titre que les tables de décision en sont un pour les procédures combinatoires;
- s'inspirer des outils de synthèse des procédures séquentielles en logique câblée;

nous avons cherché et trouvé un nouvel outil de synthèse : la table de décision séquentielle.

CHAPITRE II

CONCEPT DE LA TABLE DE DECISION SEQUENTIELLE, OUTIL DE MISE EN PAGE ET DE SIMPLIFICATION DES PROCEDURES SEQUENTIELLES

Comme signalé au chapitre précédent, nous cherchons un outil de synthèse pour les procédures séquentielles à implémenter en logique enregistrée c'est-à-dire un outil de

- mise en page
- analyse
- simplification
- préparation à la programmation

de ces procédures.

Le premier objectif annoncé est la mise en page, l'outil de synthèse cherché devra donc être capable de traduire la nature séquentielle des procédures c'est-à-dire qu'on devra pouvoir y enregistrer l'ordre de succession des opérations qui les composent.

L'idée nous est alors venue d'aller examiner ce qui se passait en hardware pour des cas semblables.

2. 1. Origine hardware.

Nous nous sommes posé la question : quelles méthodes et quels outils les gens du hardware utilisent-ils pour synthétiser leurs systèmes séquentiels ?

2. 1. 1. Synthèse des systèmes séquentiels en logique câblée.

2. 1. 1. 1. Rappel de quelques notions.

Rappelons pour commencer quelques notions de base :

a. La caractéristique d'un circuit séquentiel est le fait que l'excitation de ses bornes de sortie dépend non seulement de combinaisons de potentiels appliqués aux bornes d'entrée mais également de l'ordre selon lequel ces potentiels ont été appliquées. C'est pourquoi il est nécessaire de mémoriser les opérations et introduire des mémoires secondaires ou auxiliaires de façon à distinguer chronologiquement des combinaisons de potentiels identiques aux bornes d'entrée.

b. Un système séquentiel est entièrement défini par 2 systèmes d'équations logiques :

$$(1)\quad \begin{cases} P = f_1\,(a,\ b,\ \ldots y,\ z) \\ Q = f_2\,(a,\ b,\ \ldots y,\ z) \\ \vdots \\ Y = f^{n-1}\,(a,\ b,\ \ldots y,\ z) \\ Z = f^{n}\,\ \ (a,\ b,\ \ldots y,\ z) \end{cases}$$

et

$$(2)\quad \begin{cases} S_1 = g_1\,(a,\ b,\ \ldots y,\ z) \\ \vdots \\ S_k = g_k\,(a,\ b,\ \ldots y,\ z) \end{cases}$$

dans lesquels (P, Q, ... Y, Z) sont n mémoires secondaires introduites dans le système afin que les sorties ($S_1 \ldots S_k$) désirées, soient réalisées lorsque les bornes d'entrée (a, b, ...) sont excitées suivant les combinaisons de potentiels définies par l'énoncé du problème et se succédant dans le temps.

c. Le but de la synthèse d'un circuit séquentiel consiste à déterminer les 2 systèmes d'équations logiques afin de pouvoir passer à l'implémentation c'est-à-dire à la conception et l'élaboration des circuits dans la technologie choisie.

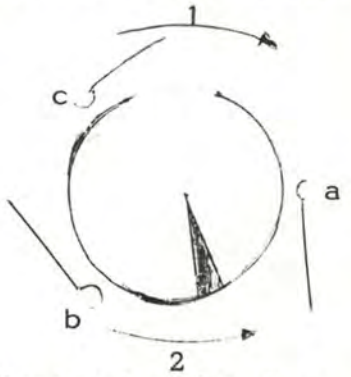
2. 1. 1. 2. Méthode de Huffman.

Il existe en logique câblée plusieurs méthodes de synthèse des circuits séquentiels. Parmi elles la méthode de Huffman. Elle est adaptée à la synthèse de tout processus séquentiel, qu'il soit simple (une séquence définie) ou complexe (plusieurs séquences définies). Nous nous y sommes intéressés parce que

1. elle utilise l'outil tabulaire;
2. sous une forme qui permet d'y enregistrer aisément les séquences définies par l'énoncé du problème;
3. que les étapes de synthèse qu'elle définit peuvent être transposées pour la synthèse des procédures séquentielles en logique enregistrée.

Nous détaillerons les différentes étapes de la méthode à partir d'un exemple afin d'assurer une compréhension plus efficace.

"Considérons un disque en matière isolante portant sur une de ses faces un petit secteur métallique étroit. Durant sa rotation, trois balais a, b, c décalés entre eux de 120°, frottent l'un après l'autre sur la surface conductrice de sorte que les



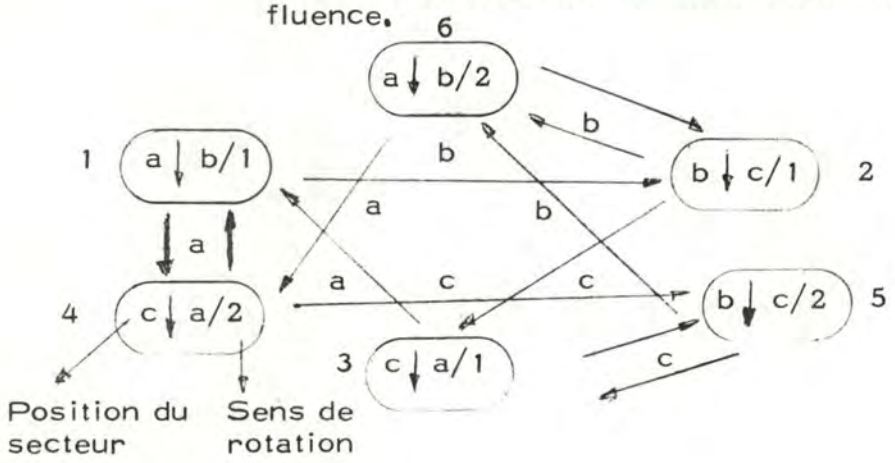
contacts s'établissent dans l'ordre a, b, c, a ou c, b, a, c selon le sens de rotation. Il s'agit de concevoir un circuit permettant de visualiser le sens de rotation au moyen de 2 lampes S_1 et S_2".

Remarque :

Nous avons choisi un exemple simplifié où la commande du circuit se fait par impulsions. Nous aurions pu choisir un exemple avec commande du circuit par niveau mais cela nous aurait amené à introduire des notions supplémentaires de hardware dont nous n'avons que faire puisqu'elles ne correspondent à rien d'équivalent en software.

La méthode de Huffman consiste à effectuer, dans l'ordre, les étapes suivantes :

1. à partir de l'énoncé ou d'une description du système séquentiel à synthétiser, effectuer le dénombrement des états qu'il peut prendre. Cette première étape peut être menée soit par raisonnement, soit en établissant un diagramme des transitions encore appelé diagramme de fluence.



Sens de rotation 1 :
... abca ...

Sens de rotation 2 :
... acba ...

Dans notre exemple, l'état du système à un moment donné est défini par :

1. le sens de rotation du disque,
2. la position du secteur.

Nous dénombrons donc 6 états. Le diagramme donne les transitions entre états du système; lorsque se présente une impulsion, le système transite dans un nouvel état fonction de l'ancien et de l'impulsion qui s'est présentée.

2. Constituer la table des phases primitive. Cette table contient toutes les informations "entrée(impulsions) - sortie (lampes S_1, S_2)" des séquences possibles du système. Elle s'établit de la manière suivante : à chaque état est associée une phase de la table et chaque phase occupe une ligne de celle-ci. Le saut d'une ligne à l'autre ou d'une phase à l'autre, s'opère à chaque impulsion. Les impulsions (d'une manière plus générale, les valeurs possibles de variables d'entrée) sont spécifiées en colonne; les transitions dans la table elle-même. Chaque ligne de la table comporte en outre l'indication des valeurs correspondantes des variables de sortie (S_1 et S_2).

Table des phases primitive :

		Impulsions a	b	c	S_1	S_2
PHASES ou ETATS	1	4	2	-	1	0
	2	-	6	3	1	0
	3	1	-	5	1	0
	4	1	-	5	0	1
	5	-	6	3	0	1
	6	4	2	-	0	1

Les transitions qui ne présentent jamais sont marquées par un tiret; il est bon de vérifier qu' elles ne correspondent pas à des omissions ou à un manque d'information sur le système.

3. Rechercher les états équivalents. Sont équivalents des états qui d'une part, admettent les mêmes sorties et d'autre part, conduisent à des états identiques ou eux-mêmes équivalents pour les mêmes combinaisons des variables d'entrée.

Dans l'exemple que nous avons choisi, il n'en existe pas; observons cependant que les états 3 et 4 conduisent à des états identiques mais qu'ils diffèrent par les valeurs des variables de sortie S_1 et S_2.

4. Réduire la table des phases primitive en fusionnant les lignes qui correspondent à des états équivalents. On obtient ainsi la table des phases réduite.

5. Organiser les variables secondaires et coder les différents états. Ayant 6 états à distinguer, 3 mémoires binaires seront nécessaires.

6. Etablir la matrice des excitations secondaires qui conduit à l'expression des fonctions d'excitation (système d'équations logiques (1)). Le codage des états étant connu, elle s'obtient en remplaçant les références des phases par les combinaisons binaires d'excitation.

   Etablir la matrice de sortie qui permettra d'établir le système d'équations logiques (2) correspondant aux sorties. Elle s'établit en remplaçant les références des phases par les combinaisons binaires de sortie.

2. 1. 2. Passage de la logique câblée à la logique enregistrée.

Une procédure séquentielle en logique enregistrée, est, rappelons-le, caractérisée par le fait que son résultat est fonction d'un ordre d'examen des conditions et/ou d'exécution des décisions/traitements. Pour pouvoir rendre compte de l'ordre de succession des conditions et décisions/traitements lors du déroulement de la procédure, nous mémoriserons ses états, comme c'est le cas pour les procédures séquentielles en logique câblée (mémorisations des états par introduction de mémoires auxiliaires de codage : étape 5 de la synthèse).

L'évolution d'une procédure séquentielle à partir d'un état quelconque de départ est donc fonction de, et complétement définie par, 2 paramètres :

a. le paramètre ETAT (E) : initialement sa valeur est celle de l'état de départ pour ensuite se modifier au cours du déroulement de la procédure comme les états qui se succèdent.

b. le paramètre CONDITION(C) : ce paramètre varie comme les conditions qui articulent la procédure au cours de son déroulement.

Remarquons que les 2 paramètres caractérisent toute procédure séquentielle, peu importe la logique dans laquelle elle doit être implémentée (en logique câblée le paramètre CONDITION est appelé ENTREE).

L'outil de synthèse cherché doit assurer la mise en page des procédures séquentielles; ces 2 paramètres devront donc y figurer.

La méthode de Huffman et l'outil de synthèse qu'elle utilise, nous ont fortement inspiré dans notre recherche. Nous lui avons en définitive empruntée l'outil tabulaire et la manière de réaliser une synthèse, pour les redéfinir et les adapter à la logique enregistrée.

L'outil tabulaire redéfini sera la table de décision séquentielle. Quant à la synthèse, nous reprendrons les mêmes étapes mais adaptées au nouvel environnement. Une légère différence se manifestera cependant en ce sens qu'en logique câblée on commence par dénombrer les états de la procédure pour leur associer ensuite une phase dans la table; en logique enregistrée, nous dénombrerons d'abord les phases qui composent la procédure, chacune d'elle définissant ensuite un état caractéristique de la procédure. Les deux démarches reviennent au même. Brièvement les étapes consisteront à :

1. rechercher les phases qui composent la procédure. Nous demandons au lecteur de patienter quelque peu, nous définirons exactement la notion de phase au paragraphe suivant.

2. établir la table de décision séquentielle qui est l'enregistrement de la procédure. Elle se présente comme les tables relatives à la synthèse des circuits en tenant compte du parallélisme entre les termes consacrés de l'une et l'autre logique :

câblée	⟷	enregistrée
entrée		condition
sortie		traitement/décision
mémoires auxiliaires		aiguillage

3. rechercher les phases équivalentes.

4. Fusionner les phases équivalentes afin de réduire la procédure qui y est enregistrée.

5. organiser les aiguillages afin de coder les différents états de la procédure.

6. préparer la programmation en établissant éventuellement un organigramme.

Les différentes étapes seront détaillées aux paragraphes suivants.

Remarque :

En suivant pas à pas les étapes de synthèse de la méthode de Huffman, nous constatons que nous ne reprenons que 3 points de la synthèse telle que nous l'avons définie au départ; à savoir :

- mise en page
- simplification
- préparation de la programmation.

Il y manque la partie analyse. L'analyse des circuits séquentiels est réduite à un minimum; tout au plus consiste-elle à examiner les tirets de la table et à se demander ce qui se passerait si les combinaisons des variables d'entrée correspondant à la colonne du tiret, se présentaient alors que le système est dans l'état correspondant à la ligne de ce tiret : s'agit-il d'un cas omis ou d'un manque d'information du système ?
En logique enregistrée, nous désirons lors de la synthèse, analyser plus profondément la procédure; l'analyser elle (séquences qu'elle détermine,boucles, cas omis, erreurs ...) et sa mise en page.

2. 2. Description de la table.

Forts de ce que nous avons découvert en hardware, nous allons à présent nous en détacher pour définir des notions propres à la logique enregistrée.

2. 2. 1. Définition de la notion de phase et d'état.

Toute procédure séquentielle est décomposable en un certain nombre de phases.

Nous définissons une phase comme l'unité d'action séquentielle qui lorsqu'elle est "en cours" peut donner lieu à la prise d'une décision ou l'exécution d'un traitement et qui ensuite donne la main à une autre phase soit conditionnellement, soit inconditionnellement.
Cette nouvelle phase alors acquiert le statut "en cours" et celle qui lui a donné la main le statut "hors cours".
Une phase peut donc correspondre :

- à la prise d'une décision ou l'exécution d'un traitement suivi d'un branchement inconditionnel à la phase suivante;

- à la prise d'une décision ou l'exécution d'un traitement suivi du test d'une condition; le résultat du test permet de déterminer la phase suivante;
- à un branchement inconditionnel sans plus;
- au test d'une condition afin de déterminer la phase suivante sans réalisation, au préalable, d'aucun traitement ni décision.

Les phases et leur succession rendent compte du comportement dynamique de la procédure lors de son déroulement.
Initialement, toutes les phases qui composent la procédure sont hors cours sauf la phase initiale et à tout instant de son déroulement, une et une seule phase est en cours.
Les procédures n'admettent jamais qu'une phase initiale mais peuvent admettre une ou plusieurs phases finales "STOP" ou "FIN" qui signifient que la procédure est terminée.

Exemple :

Soit la procédure séquentielle exprimée sous forme de l'organigramme II-1 et relatif au déplacement d'une voiture depuis sa sortie de garage jusqu'à une zone de stationnement.

Les différentes phases qui composent cette procédure sont numérotées de ① à ⑨ ; nous aurions pu les identifier par n'importe quels autres identificateurs.
Nous remarquons que toutes les phases lorsque le contrôle leur est donné et qu'elles sont en cours, donnent lieu à la réalisation d'une action avant de passer la main à la phase suivante. Les phases 2, 4 et 7 sont les seules à passer la main à la phase suivante de manière inconditionnelle.

C'est grâce à la notion de phase que nous pourrons enregistrer les opérations qui composent les procédures (test de condition et réalisation de traitements/ décisions) dans leur ordre de succession. En effet, la décomposition d'une procédure en phases permet de traduire l'ordre qui affecte les conditions et/ou décisions, par l'enchaînement des phases. Ainsi, par exemple, si deux décisions doivent se succéder dans un ordre déterminé, il suffit d'agir de façon telle à leur attribuer à chacune une phase; l'ordre sera traduit par le fait que c'est la phase associée à la première décision qui donnera la main inconditionnellement à celle associée à la deuxième.

C'est grâce aussi à la notion de phase que nous pourrons mémoriser les états de la procédure afin de lui permettre de se dérouler correctement. En effet, chaque phase définit un état de la procédure. Lorsqu'une phase a la main, la procédure se trouve dans l'état qu'elle définit. Ceci est une autre manière d'exprimer le fait que la notion de phase permet de traduire l'ordre qui caractérise une procédure séquentielle.
Les notions d'état et de phase sont distinctes mais intimement liées. Au point de vue terminologie, nous parlerons de phase ou d'état "en cours" ou "présent(e)" et encore de phase ou d'état "suivant(e)".
Une remarque : la décomposition d'une procédure en phases n'est pas unique.

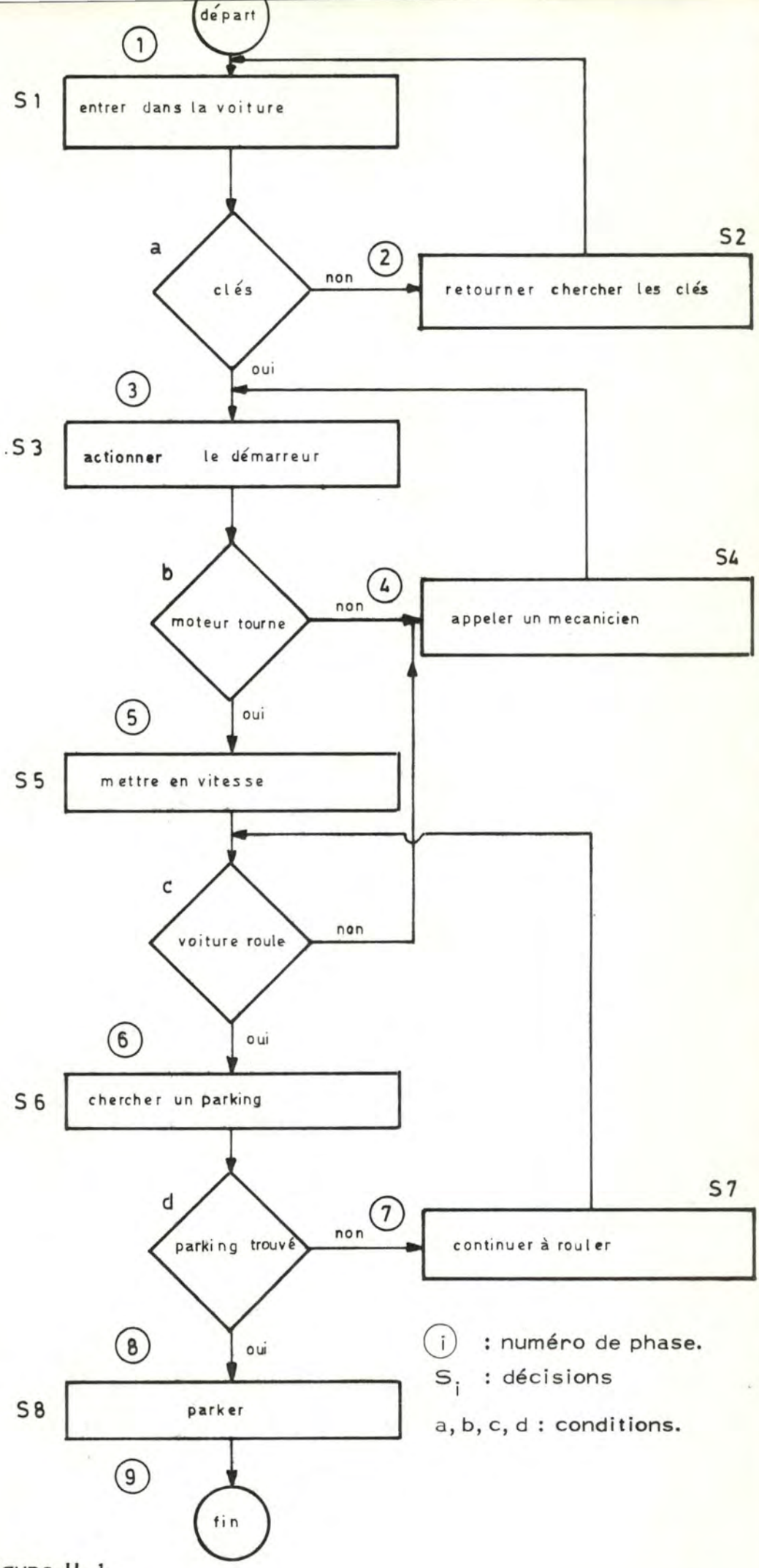

départ
1
S1
entrer dans la voiture
a
clés
non
2
S2
retourner chercher les clés
oui
3
S3
actionner le démarreur
b
moteur tourne
non
4
S4
appeler un mecanicien
oui
5
S5
mettre en vitesse
c
voiture roule
non
oui
6
S6
chercher un parking
d
parking trouvé
non
7
S7
continuer à rouler
oui
8
S8
parker
9
fin
(i) : numéro de phase.
Si : décisions
a, b, c, d : conditions.

Figure II-1.

2. 2. 2. Définition et structure de la table.

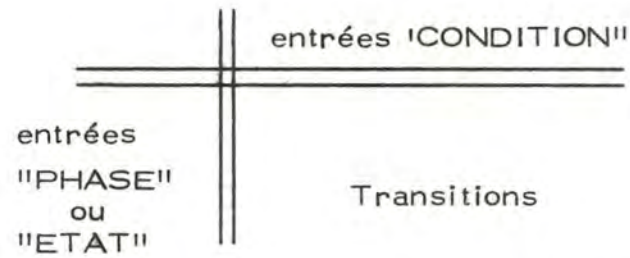
Nous allons définir la table de décision séquentielle grâce à laquelle nous pourrons en premier lieu, mettre en page les procédures.

Nous avons vu au paragraphe 2. 2. 1. comment décomposer une procédure séquentielle en phases afin de traduire l'enchaînement parfois complexe de ses décisions et traitements, subordonnés à des conditions. Une procédure séquentielle est entièrement définie par les phases qui la composent et par leur enchaînement. La table répondra donc à ce qu'on attend d'elle si :

1. elle fait apparaître ce qui compose chaque phase, c'est-à-dire quel traitement ou décision est exécuté et quelle condition est testée lorsqu'elle est en cours.

2. elle rend compte de l'enchaînement de phases qui se succédent lors du déroulement de la procédure.

La table de décision séquentielle s'inspire de, et se présente comme, la table de synthèse utilisée en hardware par la méthode de Huffman. Elle est un tableau à double entrée :



- les entrées en ligne "PHASE" ou "ETAT" : à chaque phase de la procédure est associée une ligne de la table.
  Chaque ligne, en regard de la phase qui y correspond, spécifie le traitement ou la décision qui sera éventuellement exécuté lorsque cette phase est en cours.

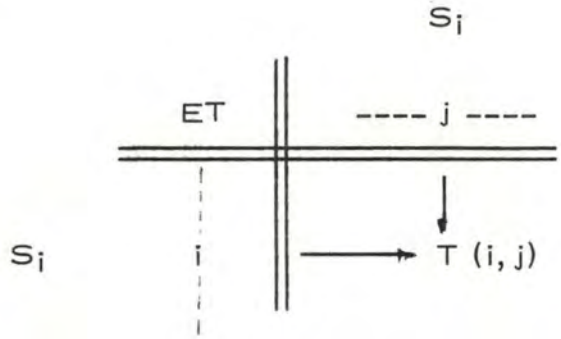
- les entrées en colonne "CONDITION" : elles reprennent toutes les conditions qui articulent la procédure. Comme une phase peut donner le main à une autre phase de manière inconditionnelle, parmi ces entrées figure celle correspondant à l'absence de condition testée : "B. I. " (branchement inconditionnel).

- le corps du tableau : il contient des identificateurs de phases (généralement des références numérotées) et rend compte de leur enchaînement. Etant donnée une phase présente, après exécution du traitement qui lui est éven-

tuellement associé, l'intersection de sa ligne et de la colonne correspondant à la condition qui testée est vérifiée, détermine la phase suivante.

Une table de décision séquentielle définit donc des relations du type :

Table T.



Si la phase présente est i ET Si, après exécution du traitement qui lui est éventuellement associé, la condition vérifiée est j, ALORS la phase suivante sera T (i, j).

Exemple :

Reprenons l'exemple du paragraphe 2. 2. 1. ; la procédure décrite est enregistrée dans la table suivante :

		Décisions	B. I.	a		b		c		d	
				0	1	0	1	0	1	0	1
PHASES	1	S_1		2	3						
	2	S_2	1								
	3	S_3				4	5				
	4	S_4	3								
	5	S_5						4	6		
	6	S_6								7	8
	7	S_7						4	6		
	8	S_8	FIN								

Conclusion :

Lors du déroulement de la procédure, on saute d'une ligne à l'autre de la table comme les phases qui se succèdent. La composition et l'enchaînement des phases apparaissent clairement dans la table.

En effet, lorsque au cours du déroulement de la procédure, une phase reçoit la main, nous savons :

1. quel traitement ou décision sera éventuellement exécuté;
2. quelle condition est éventuellement testée;
3. en fonction du résultat, quelle phase recevra la main.

Remarque :

Tout ce que nous avons exprimé en termes de phase peut être réexprimé en terme d'état de la procédure .

2. 2. 3. Paramètres de la table.

Comme annoncé, nous retrouvons dans la table, les 2 paramètres caractérisant toute procédure séquentielle:

1. le paramètre d'état E : il apparaît en ligne.

2. le paramètre de conditions C : il apparait en colonne. Notons qu'il peut être de 2 types :

   - intrinsèque: si la condition testée a été déterminée par et durant l'évolution de la procédure (dans l'exemple du paragraphe 2. 2. 1. les conditions b et c);
   - extrinsèque : si la condition testée a été fixée et acquise au départ une fois pour toutes (dans l'exemple du paragraphe 2. 2. 1. les conditions a et d).

Il est possible qu'apparaisse dans la table un 3ème paramètre : le paramètre S ou de discrimination de séquence. Il se manifeste sous forme d'aiguillages et apparait lorsque lors de l'enregistrement de la procédure, certaines phases qui la composent sont fusionnées alors qu'elles ne sont pas équivalentes. Nous anticiperons ici quelque peu en donnant la définition de phases équivalentes "deux phases sont équivalentes si elles déterminent le même traitement à exécuter pour autant qu'elles en déterminent un, et conduisent soit inconditionnellement, soit conditionnellement par test de la même condition d'entrée, à des phases elles-mêmes équivalentes ou identiques".
Ce fusionnement de phases non équivalentes a pour conséquence qu'il n'est plus possible en toute généralité de déterminer l'enchaînement des phases par simple connaissance de la phase présente et du résultat du test de la condition d'entrée; il faut dans certains cas, lorsque la phase présente a été fusionnée avec d'autres non équivalentes, connaître également la sous-séquence de la procédure qui est en train de se dérouler.

C'est pourquoi les sous-séquences de la procédure pour lesquelles il est nécessaire à certains moment de savoir si ce sont elles qui sont en train de se dérouler, seront distinguées par des aiguillages de discrimination de séquence.
Intuitivement ceci signifie, que lors du déroulement de la procédure, pour savoir où l'on va, c'est-à-dire quelle phase va recevoir la main, il faut parfois se rappeler d'où on vient.
Ce sont les aiguillages et leurs combinaisons qui permettront de connaître, lorsque nécessaire, la sous-séquence qui est en train de se dérouler.
Remarquons que le paramètre S ne dépend pas de la procédure elle-même mais de la manière dont elle est enregistrée dans la table.

Exemple :

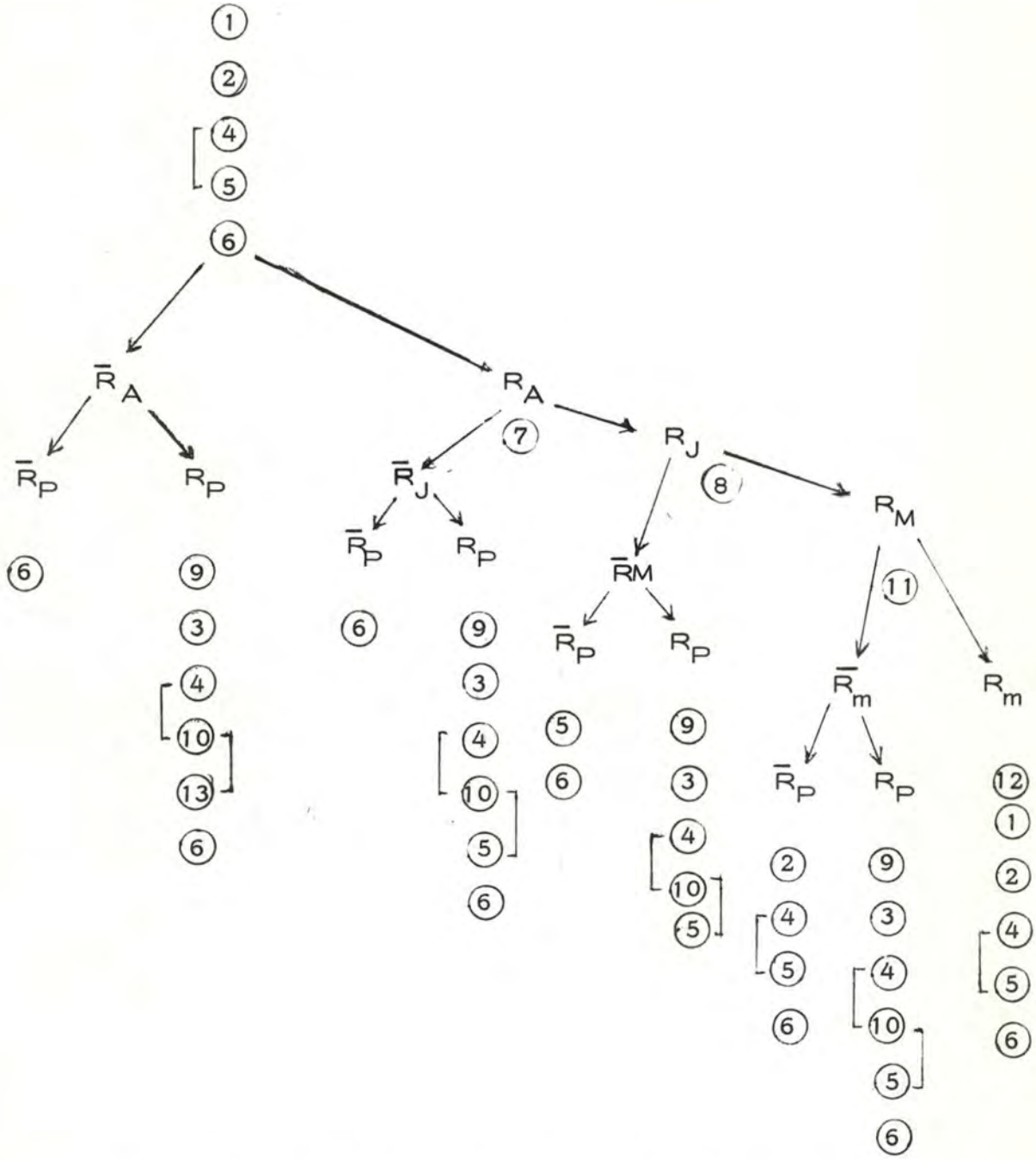
" Il s'agit d'une procédure d'édition des recettes annuelles d'une chaîne de magasins, caractérisée par 13 types de lignes d'impression différentes et qui apparaissent sur le listing dans un ordre bien déterminé en fonction des 5 conditions :

RA rupture d'article
RJ rupture de jour
RM rupture de mois
Rm rupture de magasin
RP rupture de page.

Les lignes sont les suivantes, caractérisée chacune par une référence numérotée :

1 MAGASIN .....
2 MOIS .....
4 DATE N°ART. N° ID. QUANT PRIX
5 JJMM
6 ----------
(recette de l'article pour le jour, le mois, le magasin)
7 TOTAL ARTICLE .....
8 TOTAL JOUR .....
11 TOTAL MOIS .....
12 TOTAL MAGASIN .....
9 TOTAL PAGE n° .....
3 MOIS (SUITE) .....
10 REPORT PAGE .....
13 REPORT ARTICLE .....

La succession des lignes sur le listing est donnée par le schéma suivant :



Remarquons que cette procédure est caractérisée par un ordre qui affecte uniquement les traitements.
Nous désirons enregistrer cette procédure dans une table détaillée en associant une phase à chaque type de ligne, c'est-à-dire en fusionnant toutes les phases correspondant à l'impression de la même ligne, même si elles ne sont pas équivalentes. C'est la raison pour laquelle, afin d'assurer la succession correcte des différentes lignes sur le listing par la succession des phases, il faut introduire des aiguillages de discrimination de séquence. Nous constatons que la ligne 10 peut être suivie de la ligne 13 ou 5 et que la ligne 4 peut être suivie de la ligne 10 ou 5 ; deux aiguillages S_1 et S_2 seront donc

	PHASES	DECISIONS	B. I.	0^{RA}_{1}	0^{RP}_{1}	0^{RJ}_{1}	0^{RM}_{1}	0^{Rm}_{1}	$0^{S_1}_{1}$	$0^{S_2}_{1}$
Magasin	1	ligne 1	2							
Mois	2	ligne 2;S_1=1	4							
Mois (suite)	3	ligne 3	4							
En-tête article	4	ligne 4							10 5	
En-tête JJMM	5	ligne 5	6							
Article	6	ligne 6		6' 7						
	6'	S_2 = 1 ; S_1 = 0			6 9					
Total article	7	ligne 7;S_1=0;S_2=0				7' 8				
	7'				6 9					
Total jour	8	ligne 8					8' 11			
	8'				5 9					
Total page	9	ligne 9	3							
Report page	10	ligne 10								5 13
	11	ligne 11							11' 12	
	11'				2 9					
Total magasin	12	ligne 12	1							
Report article	13	ligne 13	5							

Figure II-2

nécessaires, caractérisant l'un les sous-séquences R_P et $\bar{R}_P$, l'autre les sous-séquences R_A et $\bar{R}_A$.
La table obtenue est celle de la figure II-2.
Le bon ordre de succession des lignes sera donc assuré par les 3 paramètres : E-C-S.
Remarquons que le paramètre S est intrinsèque, en effet, S_1 et S_2 sont positionnés durant le déroulement de la procédure.

2. 2. 4. Problème des mémorisations d'états.

Nous avons annoncé au paragraphe 2. 1. 2. que pour rendre compte de l'ordre qui affecte les procédures séquentielles, afin de pouvoir à tout moment décider des opérations à réaliser, on introduisait des mémorisations d'état, tout comme c'est le cas pour les systèmes séquentiels en logique câblée.

C'est pourquoi lors de la synthèse, l'avant dernière étape consiste à coder les différents états ou phases du système par des mémoires auxiliaires qui en logique enregistrée prennent la forme d'aiguillages (variables particulières, généralement binaires). On peut caractériser chaque phase de la procédure soit par un aiguillage qui lui est propre, soit par une combinaison d'aiguillages (dans ce cas, le nombre d'aiguillages nécessaire sera réduit).

La mise à jour des aiguillages, dans le cas, par exemple, d'un aiguillage binaire par phase, consiste en ceci : lors d'un changement de phase, l'aiguillage caractéristique de la phase qui était en cours est remis à "0", tandis que l'aiguillage caractéristique de la phase qui reçoit la main est mise à "1".

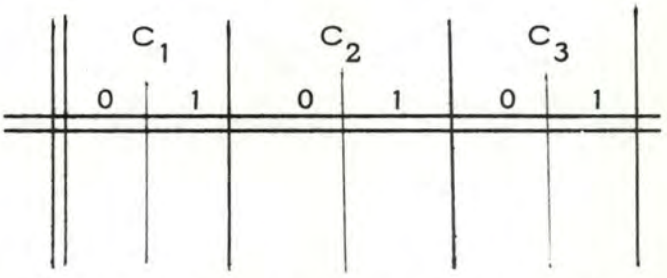
Mais il apparait que les aiguillages sont superflus en logique enregistrée. En effet, les langages de programmation tels qu'ils sont conçus et le fait que tout processeur central dans un système d'exploitation, possède un registre particulier d'adresse d'instruction qui à tout instant du déroulement de la procédure implémentée indique le point atteint, permettent de s'en passer. Ils sont implicitement définis et déterminés.

Dans certains cas, néanmoins, les aiguillages ne sont pas totalement inutiles : lorsque l'utilisateur désire jalonner la situation logique de la procédure lors de son déroulement afin d'examiner plus facilement les cas d'anomalie ou d'erreur et d'avoir des points de reprise.

2.3. Type de table : détaillé - condensé - mixte.

Les conditions que nous trouvons en en-tête des tables et qui référencent leurs colonnes, peuvent y figurer de deux manières différentes donnant naissance à deux types de table différents : les tables détaillées et les tables condensées.

1. Les tables détaillées : les conditions figurent en en-tête de manière générale en ce sens que toute condition Ci est introduite sur deux colonnes : l'une est caractérisée par un "zéro" booléen et correspond à sa valeur fausse; l'autre est caractérisée par "un" booléen et correspond à sa valeur vraie.



En chaque phase, une seule condition est testée et selon qu'elle est ou non vérifiée, la procédure branchera à une phase ou à une autre.
La table sous cette forme possède un caractère dichotonomique.

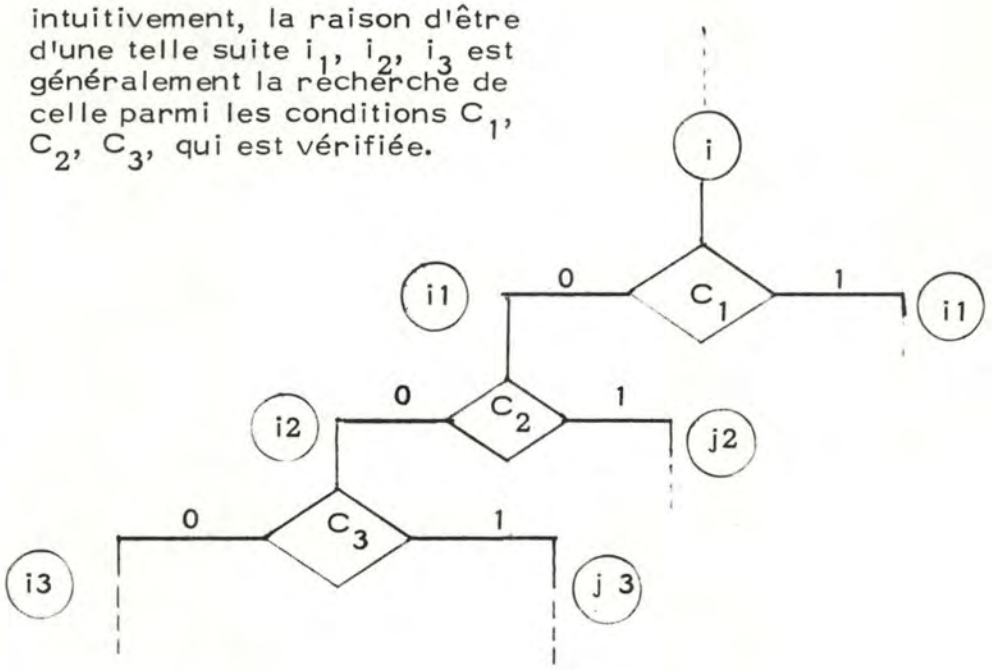
2. Les tables condensées : elles sont caractérisées par le fait que :

   a. il n'y a que les valeurs significatives des conditions qui sont prises en considération, autrement dit, les conditions ne se voient attribuer qu'une seule colonne : celle qui correspond à leur valeur vraie.

   b. certaines phases de la procédure y sont enregistrées implicitement :

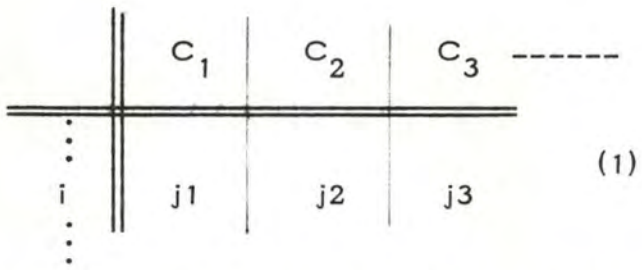
      Si une phase i donne la main à une suite de phases i1, i2, i3 ... qui se succèdent conditionnellement par test négatif et qui ne spécifient, aucune, un traitement à exécuter.

intuitivement, la raison d'être d'une telle suite i_1, i_2, i_3 est généralement la recherche de celle parmi les conditions C_1, C_2, C_3, qui est vérifiée.



ALORS les phases de cette suite seront enregistrées implicitement.

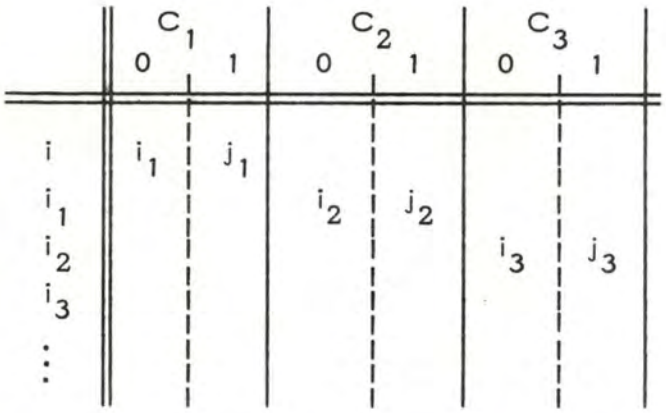
Soit la ligne de la table associée à la phase i :



(1)

Son interprétation est la suivante : lorsque la phase i est en cours, après exécution du traitement qu'éventuellement elle détermine, tester les conditions les unes après les autres de gauche à droite : si la 1ère condition (C_1) est vérifiée, la phase suivante est j1; sinon tester la 2ème (C_2), si elle est vérifiée la phase suivante est j2; sinon .....).
Dans le cas particulier où il n'y a jamais qu'une condition au plus qui est vérifiée parmi toutes les conditions à tester en cascade, la ligne peut aussi s'interprêter comme suite : lorsque la phase a terminé l'exécution du traitement qu'éventuellement elle détermine, la phase suivante à laquelle elle donnera la main est déterminée par celle des conditions qui est vérifiée.

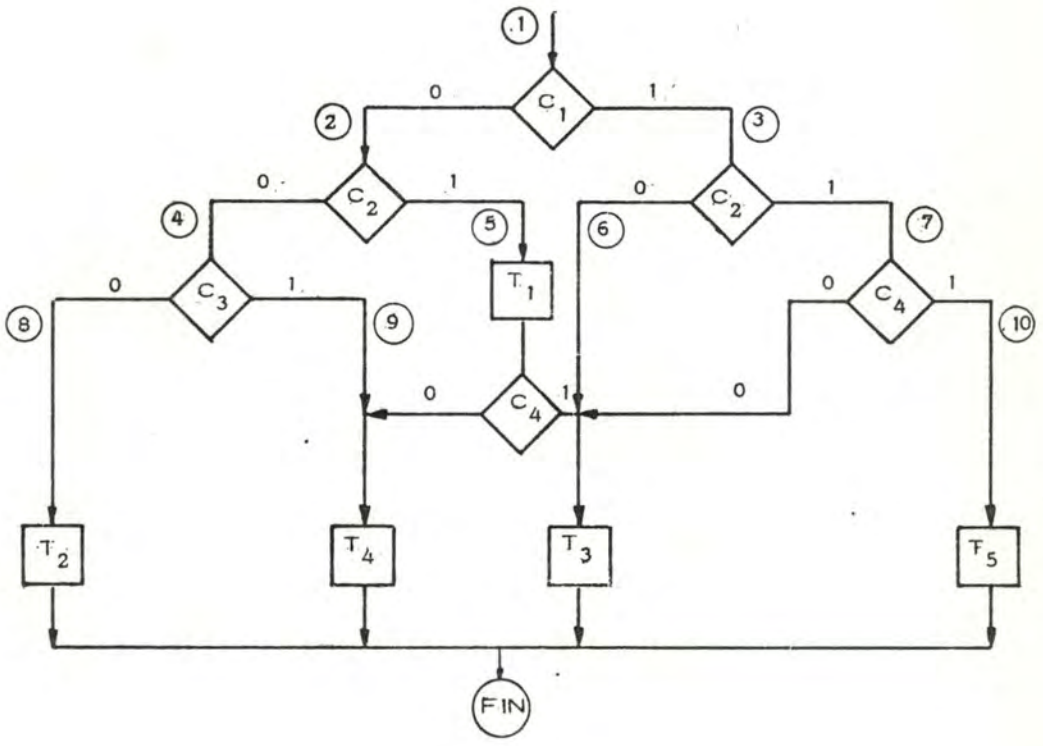
Autrement dit (1) est équivalent à :



Il faut, dans la plupart des cas, introduire dans la table une colonne supplémentaire afin de pouvoir déterminer quelle phase sera phase suivante lorsqu'aucune des conditions testées n'est vérifiée. ($\overline{C}_i$)

Exemple :

Considérons l'organigramme suivant.



et sa présentation sous forme de table détaillée et condensée :

PHASES	TRAIT.	B. I.	C_1 0	C_1 1	C_2 0	C_2 1	C_3 0	C_3 1	C_4 0	C_4 1
1			2	3						
2					4	5				
3					6	7				
4							8	9		
5	T_1								9	6
6	T_3	FIN								
7									6	10
8	T_2	FIN								
9	T_4	FIN								
10	T_5	FIN								

Table détaillée

PHASES	TRAIT.	B. I.	C_1	C_2	C_3	C_4	$\overline{C}_i$
1			3	5	9		8
3				7			6
5	T_1					6	9
6	T_3	FIN					
7						10	6
8	T_2	FIN					
9	T_4	FIN					
10	T_5	FIN					

Table condensée

Cas particulier :

Il existe des procédures qui se prêtent particulièrement bien à un enregistrement dans une table condensée. Il s'agit de procédures dont les décisions à prendre ou traitements à exécuter sont fonction de l'ordre et de l'organisation de symboles lus ou apparaissant successivement. Le test des conditions consiste dans ce cas, à vérifier quelle valeur a le symbole lu ou le caractère qui est apparu (par exemple : procédures d'échanges en télétraitement, décodage de messages, analyse syntaxique ....).

3. Les tables mixtes : il s'agit de tables dont certaines lignes et colonnes sont à interprêter comme celles des tables détaillées, d'autres comme celles des tables condensées.

Conclusion.

Toute procédure peut toujours être enregistrée dans une table détaillée. Cette forme de table est très proche de l'organigramme, la relation qui les lient est évidente.
Si certaines phases de la procédure satisfont la propriété énoncée plus haut, il y a tout intérêt à enregistrer cette procédure dans une table condensée comme le mot l'exprime. Celle-ci, par rapport à l'organigramme est une forme beaucoup plus concise et manipulable.

Exemple :

Reprenons l'exemple de la mise à jour d'un fichier (chapitre I paragraphe 1. 1.)
Enregistrons-la dans une table détaillée :

PHASES	DECISIONS	C		A		M	
		0	1	0	1	0	1
1		2	3				
2				4	5		
4						-	6
3	"création"	2'	7				
2'				4'	5		
4'						-	6
5	"annulation"	2''	3				
2''				4''	7		
4''						-	7
6	"modification"	2'''	7				
2'''				4'''	5		
4'''						-	6
7	"erreur"						

Enregistrons-la, à présent, dans une table condensée :

PHASES	DECISIONS	C	A	M
1		3	5	6
3	"création"	7	5	6
5	"annulation"	3	7	7
6	"Modification"	7	5	6
7	"erreur"			

L'utilisation de la table de décision séquentielle en tant qu'outil de mise en page est justifiée par sa forme condensée : elle est une présentation claire et concise de la procédure, accessible à tous, informaticiens et non informaticiens.

Toutes les applications que nous traiterons au chapitre V, utilisent la table sous sa forme condensée (éventuellement mixte).

2.4. Méthode d'élaboration de tables.

L'objet de ce paragraphe est de proposer à l'utilisateur quelques méthodes d'élaboration des tables afin de l'aider à enregistrer ses procédures séquentielles. Les méthodes lui sont présentées à titre indicatif; elles ne sont pas uniques; chaque utilisateur peut avoir la sienne. Nous en profiterons également pour donner quelques exemples d'enregistrement de procédures dans une table séquentielle.

Lors de la mise en page d'une procédure dans une table séquentielle, il faut avant toute chose déterminer le type de table dans laquelle se fera l'enregistrement : détaillé-condensé-mixte. Ce choix fait, il convient de faire un inventaire des conditions qui articulent la procédure c'est-à-dire des situations différentes pouvant avoir une influence sur les traitements à effectuer :

- si la table est détaillée : introduire les conditions en en-tête de la table, chacune référencant 2 colonnes (l'une correspond à sa valeur vraie, l'autre à sa valeur fausse);
- si la table est condensée: introduire les conditions en en-tête de la table, à raison d'une condition par colonne. Dans le cas particulier de procédures où les traitements dépendent de l'ordre et de l'organisation de symbole lus

ou de caractères apparaissant successivement, il suffit d'enregistrer en en-tête les caractères ou symboles significatifs. Considérer éventuellement une colonne supplémentaire propre aux cas où aucune condition n'est vérifiée.
(Dans les deux cas, il ne faut pas oublier la colonne des branchements inconditionnels "B. I. ".)

Ensuite, pour l'élaboration de la table elle-même, on peut agir de deux manières :

- soit décomposer entièrement la procédure en phases; puis les phases et leur enchaînement étant connus, l'élaboration de la table ne doit plus poser de problèmes;
- soit élaborer la table en définissant les phases au moment même ou on les introduit dans la table.

2. 4. 1. Méthode des séquences de base.

Précisons d'abord la notion de séquence : une séquence est une suite de conditions vérifiées et de traitements exécutés les uns après les autres. Une procédure lorsqu'elle se déroule détermine une séquence particulière. Toute procédure est entièrement définie par toutes les séquences auxquelles peut correspondre son déroulement c'est-à-dire par tous les chemins qu'elle peut suivre.

Cette méthode est utilisée lorsque la procédure doit sa nature séquentielle à un ordre sur les conditions et un ordre sur les traitements. Les séquences définies par ces procédures sont :

- soit composées d'une suite de conditions qui sont vérifiées (dans certains cas particuliers d'une suite de symboles ou caractères qui doivent être lus ou apparaître successivement) suivie du ou des traitements à exécuter en conséquence.
  Les séquences de ce type sont appelées séquences de base et se déduisent de l'énoncé.

- soit composées d'une suite de conditions qui sont vérifiées et qui n'aboutissent à aucun traitement. Elles constituent une séquence impossible, indifférente ou erronée.

Exemple :

" On reçoit un message composé de caractères a, b, c. Chaque fois que l'on détecte dans l'ordre les 3 caractères abc, il faut incrémenter un compteur (traitement T1)".

Cette procédure très simple, ne possède qu'une séquence de base notée sous la forme

conditions vérifiées traitement

Etant données deux séquences de base :

- l'une peut être origine de l'autre ;

  exemple : abc ⟶ T_1 et abc ca ⟶ T_2

- elles peuvent avoir une origine commune;

  exemple : abc ⟶ T_1 et ab ba ⟶ T_2

- l'une peut être incluse dans l'autre;

  exemple : abc ⟶ T_1 et x abc y ⟶ T_2

La méthode consiste à effectuer les étapes suivantes :

1. Prélever, à partir de l'énoncé, les séquences de base de la procédure;
2. Les enregistrer dans la table (détaillée, condensée ou mixte) à partir d'une phase initiale, en définissant et en introduisant les phases au fur et à mesure : chaque phase sera composée soit d'un test de condition(s), soit d'une exécution de traitement.
   Il y a intérêt à enregistrer les séquences de base dans l'ordre suivant :
   a. celles qui n'en contiennent pas d'autres, n'ont pas d'origine commune avec d'autres et ne sont pas elle-même origine;
   b. celles qui sont origine d'une autre, prolongées par cette autre;
   c. celles qui ont une origine commune, introduire la sous-séquence commune pour ensuite l'éclater et poursuivre les enregistrements de chacune d'elles;
   d. celles qui en contiennent d'autres, sans oublier d'introduire au moment voulu les traitements déterminés par les séquences qu'elles contiennent.
3. Compléter, s'il en existe, les cases vides par des phases d'indifférence, de rejet ou encore par des phases qui sont déjà définies de façon à respecter la logique des décisions lors des ruptures des séquences de base.

Exemple :

" Une expression arithmétique, si on distingue deux niveaux d'opérateur, est exprimée dans le formalisme de Backus-Naur par :

< expression > ::= operande | < expression > opérateur niveau bas > < expression > | < expression > < opérateur niveau haut > < expression >

< operateur niveau bas > ::= + | -

< operateur niveau haut > ::= * | /

L'application consiste à jeter les bases d'un programme susceptible de prendre toutes les mesures utiles pourque l'exécution d'une expression puisse se réaliser correctement. L'exécution se fait en 2 étapes : la 1ère consiste à exécuter les opérations de niveau haut pour exécuter au cours de la 2ème, les opérations de niveau bas.
La procédure chargée de la 1ère étape devra réaliser les traitements suivants :

- la lecture d'un opérande a pour effet de le placer dans une zone Z_1 (zone qui contient toujours le dernier opérande lu ou calculé);
- la lecture d'un opérateur de niveau haut (O_1) a pour effet d'exécuter l'expression formée de l'opérande stocké dans Z_1, l'opérateur O_1 et l'opérande suivant; de placer le résultat dans Z_1;
- la lecture d'un opérateur de niveau bas (O_2) a pour effet de placer l'opérande contenu dans Z_1, dans Z_T (zone de stockage qui en fin de 1ère partie contiendra les opérandes des opérateurs de niveau bas) et l'opérateur O_2 dans une zone Z_0 (qui en fin de 1ère partie contiendra tous les opérateurs de niveau bas);
- la lecture du caractère fin d'expression a pour effet de vider une dernière fois Z_1 dans Z_T et de donner la main à la 2ème partie qui exécutera l'expression composée alternativement des opérandes de Z_T et des opérateurs de Z_0".

Examinons l'effet de la procédure sur l'expression :

a + b * c / d - e ▲

après lecture de	contenu des zones Z_1	Z_T	Z_0
a	a		
+		a	+
b	b	a	+
* c	b * c	a	+
/ d	b * c/d	a	+
-		$b *_a c/d$	$\bar{+}$
e	e	$b *_a c/d$	$\bar{+}$
		$b *^e_a c/d$	$\bar{+}$

Elaborons la table de mise en page de la procédure correspondant à la 1ère partie :

0. il y a avantage à enregistrer cette procédure dans une table condensée; en en-tête figurent les symboles lus
   - O_1 opérateur niveau haut
   - O_2 opérateur niveau bas
   - C opérande
   - ▲ fin d'expression.

1. les séquences de base sont :
   - (1) C $\longrightarrow T_1$ c'est-à-dire mettre C dans Z_1
   - (2) $CO_1C \longrightarrow T_2$ c'est-à-dire exécuter $Z_1 O_1 C$ et mettre le résultat dans Z_1
   - (3) $CO_2 \longrightarrow T_3$ c'est-à-dire mettre Z_1 dans Z_T et O_2 dans Z_0
   - (4) C ▲ $\longrightarrow T_4$ c'est-à-dire mettre Z_1 dans Z_T et brancher à la 2e partie.

2. Enregistrer:
   - la séquence de base (1) qui est origine de toutes les autres; définir et introduire à cet effet la phase 2;
   - les séquences (2),(3), (4) par prolongement de(1); définir et introduire à cet effet les phases 3 et 6; 4; 5.

PHASES	TRAIT.	B. I.	C	01	02	
1			2	(E)	(E)	(E)
2	T_1		(E)	3	4	5
3			6	(E)	(E)	(E)
4	T_3	(1)				
5	T_4	FIN				
6	T_2			(3)	(4)	(5)

3. Compléter les cases vides pour respecter la logique des décisions lors des ruptures; les cases complétées sont entre parenthèses.
La phase "E" signale que l'expression de départ est incorrecte.

Remarque :

Cette méthode sera utilisée au chapitre V pour élaborer les tables séquentielles de l'application III relative à l'analyse syntaxique du langage BASIC.

2. 4. 2. Méthode pas à pas.

Elle consiste à élaborer la table ligne par ligne, c'est-à-dire phase par phase comme suit :

1. Se poser la question : la phase correspondant à la ligne examinée détermine-t-elle un traitement à exécuter ?

   si oui : indiquer le traitement en regard de la phase;
   aller en 2;

   si non : aller en 2.

2. Si la table est détaillée, se poser la question : quelle condition parmi les conditions figurant en en- tête, est à tester pour pouvoir déterminer la suite de la sous-séquence examinée ?

   Si la table est condensée, se poser la question : quelles conditions sont susceptibles d'être vérifiées ?

   Dans les deux colonnes correspondant à la condition testée ou dans les colonnes correspondant aux conditions susceptibles d'être vérifiées, selon le cas, introduire :

   - soit la référence d'une nouvelle phase et lui associer une nouvelle ligne dans la table;

- soit la référence d'une phase déjà décrite (ce qui revient à fusionner deux sous-séquences de la procédure);
- soit la référence de la phase qui caractérise les cas d'erreur.

Les deux étapes sont à itérer jusqu'à ce qu'il n'y ait plus de nouvelle phase introduite dans la table et débutent avec la phase initiale.

Cette méthode sera utilisée pour élaborer les tables séquentielles de l'application IV du chapitre V.

Exemple :

" Soit le procédure de transposition d'une expression arithmétique de forme infixée en forme polonaise postfixée. L'expression de forme infixée est composée :

- d'opérandes
- d'opérateurs : ils sont caractérisés par des niveaux de priorité :

**	↑
x , /	priorité croissante
+ , -	

- de parenthèses "(" et ")" destinées à modifier, comme on le sait, les priorités des opérateurs.
  La parenthèse "(", par suite de l'algorithme de transposition est considérée comme opérateur de priorité la plus basse.

La procédure de transposition utilisé 2 zones de stockage :

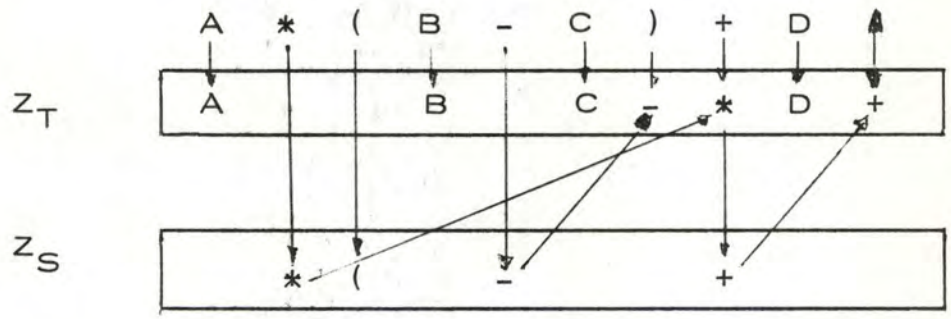
- Z_T destinée à contenir la forme postfixée
- Z_S zone intermédiaire

et consiste à réaliser les traitements suivants :

1. à la lecture d'un opérande, le placer dans Z_T
2. à la lecture d'un opérateur :
   - s'il est strictement plus prioritaire que le dernier opérateur introduit dans Z_S, le placer dans Z_S
   - sinon déplacer le dernier opérateur introduit dans Z_S, dans Z_T et recommencer le test;
3. à la lecture d'une "(" : la placer dans Z_S;

4. à la lecture d'une ")" : vider Z_S dans Z_T selon la technique LIFO jusqu'à la première "(" rencontrée; enlever cette "(";
5. à la lecture du symbole ▲ (fin d'expression) : vider Z_S dans Z_T selon la technique LIFO; la transposition est terminée ".

Examinons shématiquement la façon de procéder, sur l'exemple suivant :



En procédant comme décrit par la méthode, nous obtenons la table mixte :

Phases	TRAIT.	B. I.	Opérande	Opérateur	(	)	▲	Opérateur lu > Opérateur Z_S	
								Oui	Non
1			2	E	6	E	8		
2	opérande dans Z_T		E	3	E	7	8		
3								4	5
4	opérateur lu dans Z_S		2	E	6	E	E		
5	opérateur de Z_S dans Z_T							4	5
6	(dans Z_S		2	E	6	E	E		
7	vider Z_S dans Z_T jusqu'à (; enlever (		E	3	E	7	8		
8	vider Z_S dans Z_T	Fin							
E	erreur	Fin							

Nous avons supposé que les parenthèses "(" et ")" étaient présentes par paires dans l'expression de départ.

2. 4. 3. Méthode du graphe.

Cette méthode permet d'élaborer les tables séquentielles par l'intermédiaire d'un graphe. Elle examine la procédure à enregistrer sous l'angle des traitements et de leur succession lors du déroulement de la procédure. Elle élabore la table en 2 étapes.

Etape I.

Elle consiste à définir un graphe, généralement un multigraphe, qui rend compte de la succession possible des traitements (un multigraphe est caractérisé par le fait que deux sommets quelconques peuvent être l'un extrémité initiale et l'autre extrémité finale de plus d'un arc).

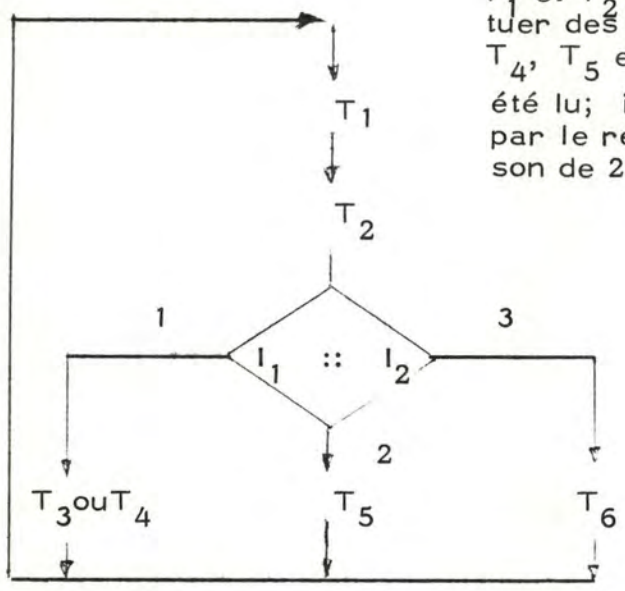
a. tout traitement qui apparait dans l'énoncé de la procédure définit un sommet du graphe;

b. un arc est défini d'un sommet T_i vers un sommet T_j, chaque fois que, lors du déroulement de la procédure, l'exécution du traitement T_i peut être suivie de celle de T_j ;

   - soit inconditionnellement : dans ce cas l'arc sera surmonté de l'indication "i";
   - soit conditionnellement : dans ce cas, l'arc sera surmonté de la ou des conditions qui vérifiées dans l'ordre, provoquent le passage d'un traitement à l'autre.

Exemple :

"Soit une procédure dont le déroulement correspond à l'exécution de 6 traitements : T_1 à T_6, selon le schéma répétitif:



T_1 et T_2 sont destinés à effectuer des lectures tandis que T_3, T_4, T_5 et T_6 traitent ce qui a été lu; ils sont conditionnés par le résultat de la comparaison de 2 indicatifs I_1 et I_2

Chaque nouvelle boucle est fonction de celle qui la précède : ainsi, après exécution de T_3, il faut exécuter T_1, mais non T_2 et ensuite selon le test des indicatifs T_4, T_5 ou T_6 ; ceci est noté sous la forme :

$$T_1 \overline{T}_2 \cdot (T_4 + T_5 + T_6)$$

de même après
$$\begin{cases} T_4 : \overline{T}_1 T_2 \cdot (T_3 + T_5 + T_6) \\ T_5 : T_1 T_2 \cdot (T_4 + T_5 + T_6) \\ T_6 : T_1 \overline{T}_2 \cdot (T_3 + T_5 + T_6) \end{cases}$$

le graphe défini par cette procédure est celui de la figure II. 3.

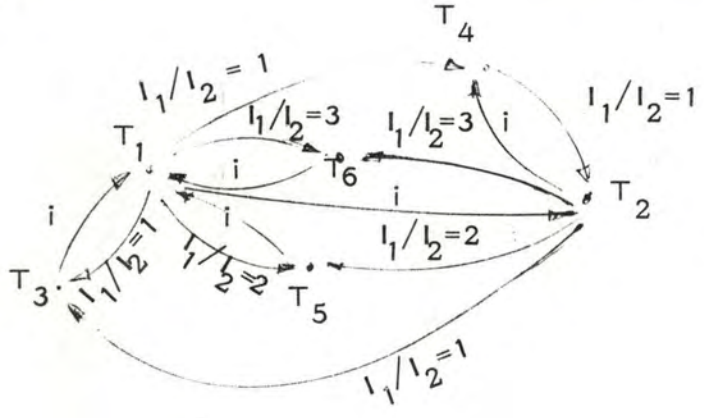


Fig. II-3

Etape II :

Elle consiste à passer du graphe à la table en associant à chaque sommet du graphe une phase, c'est-à-dire en définissant une phase par traitement. Nous détaillerons ce passage dans le cas où l'utilisateur a choisi d'enregistrer la procédure dans une table condensée. Il n'est pas difficile de modifier cette 2ème étape pour l'adapter aux tables détaillées.

1. Repérer les sommets qui sont extrémité initiale de plus d'un arc dont un est surmonté de "i" c'est-à-dire inconditionnel.
   Soit T_i un tel sommet.
   Comme nous associons une phase par traitement, ceci signifie que nous fusionnons, au niveau de la phase associée à T_i, 2 sous-séquences non équivalentes. Il faut par conséquent introduire un aiguillage de discrimination de séquences S pour se rémémorer la sous-séquence qui se déroule.
   Dans l'exemple, T_1 répond au critère énoncé, on a

$$T_1 \xrightarrow{i} T_2 \quad \text{et} \quad \begin{cases} T_1 \xrightarrow{I_1/I_2 = 1} T_3 \\ T_1 \xrightarrow{I_1/I_2 = 1} T_4 \\ T_1 \xrightarrow{I_1/I_2 = 2} T_5 \\ T_1 \xrightarrow{I_1/I_2 = 3} T_6 \end{cases}$$

   nous introduirons donc un aiguillage S_1, afin de discriminer les 2 possibilités; il sera positionné au 1 au cours de la phase associée à T_5, à 0 au cours des phases associées à T_3 et T_6.
   Au niveau du graphe, introduire un sommet artificiel T'_j qui devient nouveau sommet initial des arcs qui admettent T_i pour sommet initial, sauf pour l'arc inconditionnel.
   Le sommet T'_i est joint à T_i par l'arc $T_i \longrightarrow T'_i$, arc qui est surmonté de la condition $S = 0$ (ou (1).
   L'arc inconditionnel sera surmonté non plus de "i" mais de la condition $S = 1$ (ou 0).
   Les conditions $S_1 = 0$ et $S = 1$ s'ajoutent aux conditions figurant en en-tête de la table.

2. Repérer les arcs qui admettent même sommet initial, soit T_i, sont surmontés des mêmes conditions à vérifier mais admettent des sommets terminaux différents. Nous sommes à nouveau dans une situation identique à celle rencontrée en 1 : fusion de sous-séquences non équivalentes. Nous introduirons donc également des aiguillages de discrimination de séquences.

Dans l'exemple nous avons :

$$\begin{cases} T_1 \xrightarrow{I_1/I_2 = 1} T_3 \\ T_1 \xrightarrow{I_1/I_2 = 1} T_4 \end{cases} \quad \text{et} \quad \begin{cases} T_2 \xrightarrow{I_1/I_2 = 1} T_3 \\ T_2 \xrightarrow{I_1/I_2 = 1} T_4 \end{cases}$$

Pour les 2 cas, il suffira d'introduire un seul aiguillage S_2; positionné à "1" au cours des phases associées à T_3 et T_5 et à "0" au cours des phases associées à T_4 et T_6.

Au niveau du graphe, pour chaque cas, introduire un sommet artificiel T'_i; le joindre d'une part à T_i par l'arc $T_i \longrightarrow T'_i$ surmonté de la ou des conditions identiques, d'autre part aux sommets terminaux par des arcs surmontés des conditions qui testées permettront de déterminer la sous-séquence parcourue.

3. Associer une phase à chaque sommet du graphe, éventuellement redéfini par les étapes 1 et 2 ; enregistrer ces phases dans la table avec leur traitement ainsi que leur enchaînement connu par simple considération du graphe.

Exemple :

Le graphe de la figure II-2, a été transformé en celui de la figure II-4 par l'étape II de la méthode.

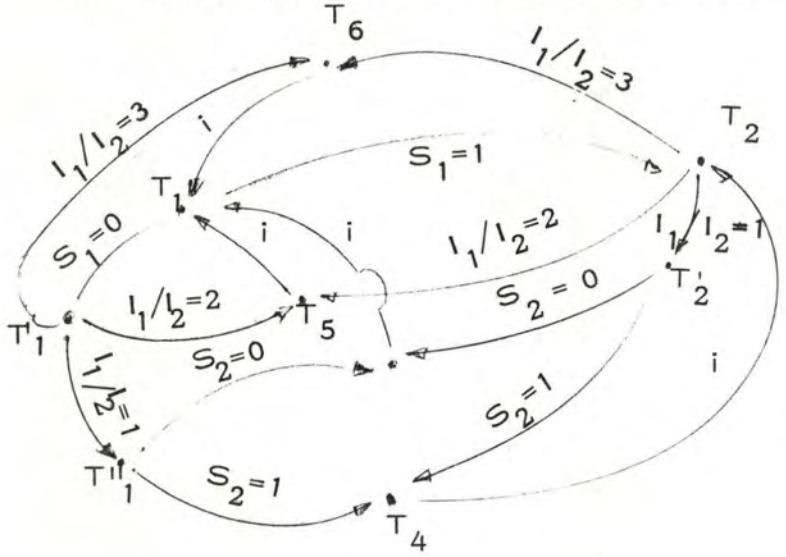


Fig. II-4

La proécure est enregistrée à partir du graphe, dans la table

Phases	TRAIT.	B. I.	I_1 / I_2			S_1		S_2	
			1	2	3	0	1	0	1
1	T_1					1'	2		
1'			1''	5	6				
1''								3	4
2	T_2		2'	5	6				
2'								3	4
3	T_3	1							
4	T_4	2							
5	T_5	1							
6	T_6	1							

Les quelques méthodes qui ont été exposées à titre indicatif, sont laissées à l'appréciation du lecteur; libre à lui d'en imaginer et d'en utiliser d'autres.

2. 5. Simplification des procédures et réduction des tables par fusion des phases équivalentes.

Nous avons, au cours des paragraphes précédents, montré comment la table de décision séquentielle permettait d'enregistrer des procédures séquentielles; le premier objectif de synthèse est par conséquent atteint. Intéressons-nous à présent à un deuxième objectif : la simplification des procédures. Nous allons donc détailler les étapes 4 et 5 de la synthèse (revoir § 2. 1. 2.).

Il se peut que lors de la décomposition d'une procédure en phases, nous ayons défini des phases qui soient équivalentes. Il est extrêmement utile de les découvrir afin de pouvoir, en les fusionnant,

a. simplifier la procédure
b. réduire la table séquentielle dans laquelle a été mise en page cette procédure.

2. 5. 1. Définition des phases équivalentes.

Deux phases sont équivalentes si :

1. elles déterminent le même traitement à exécuter;
2. elles mènent soit inconditionnellement, soit conditionnellement par vérification des mêmes conditions d'entrée, à des phases elles-mêmes équivalentes ou identiques.

Cette définition est récursive. Intuitivement, deux phases qui sont équivalentes, signifient que les deux sous-séquences de la procédure qui admettent ces phases pour phase initiale, sont identiques. Deux phases équivalentes correspondent à un seul et même état de la procédure.

Exemple :

Considérons la table séquentielle de la figure II-5; nous observons les équivalences suivantes :

3 ≡ 13
7 ≡ 15
5 ≡ 17
6 ≡ 18
2 ≡ 14
4 ≡ 16

PHASES	TRAIT.	B. I.	C_1		C_2		C_3		C_4	
			0	1	0	1	0	1	0	1
0	-		1	11						
1	-				2	3				
2	T_0						4	5		
3	R	FIN								
4	T_4								6	7
5	T_3	FIN								
6	T_1	FIN								
7	T_2	FIN								
11	-				12	13				
12	T_0								14	15
13	R	FIN								
14	T_0						16	17		
15	T_2	FIN								
16	T_4	FIN							18	15
17	T_3	FIN								
18	T_1	FIN								

Fig. II-5

2. 5. 2. Méthode de recherche systématique des phases équivalentes.

La méthode que nous allons exposer, permet de découvrir les phases équivalentes de manière systématique sans se perdre dans la récursivité de la définition. Nous l'exposerons en parallèle d'un point de vue théorique et d'un point de vuepratique sur l'exemple de la figure II-5.
Les étapes de la méthode sont les suivantes :

1. Etablir une partition sur l'ensemble des phases par la relation d'équivalence "déterminer le même traitement à exécuter". Ceci correspond à la vérification de la première condition pourqu'il y ait équivalence.

   Exemple : les classes de la partition sont :

   $-$: { 0, 1, 11 }
   T_0 : { 1, 12, 14 }
   T_1 : { 6, 18 }
   T_2 : { 7, 15 }
   T_3 : { 5, 17 }
   T_4 : { 4, 16 }
   R : { 3, 13 }

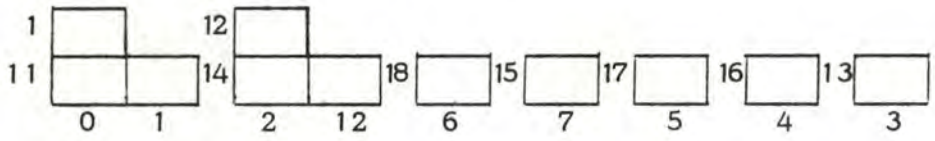
2. Pour toute classe de la partition qui possède plus d'un élément :

   a. établir un tableau en escalier :

   - toute phase, sauf la première, prise dans l'ordre dans lequel apparaît dans la classe, définit une ligne du tableau;
   - toute phase, sauf la dernière, prise dans l'ordre dans lequel elle apparait dans la classe, définit une colonne du tableau.

   Ce tableau rendra compte des équivalences entre phases, prises deux à deux.

   Exemple : les différentes classes définissent les tableaux en escalier.



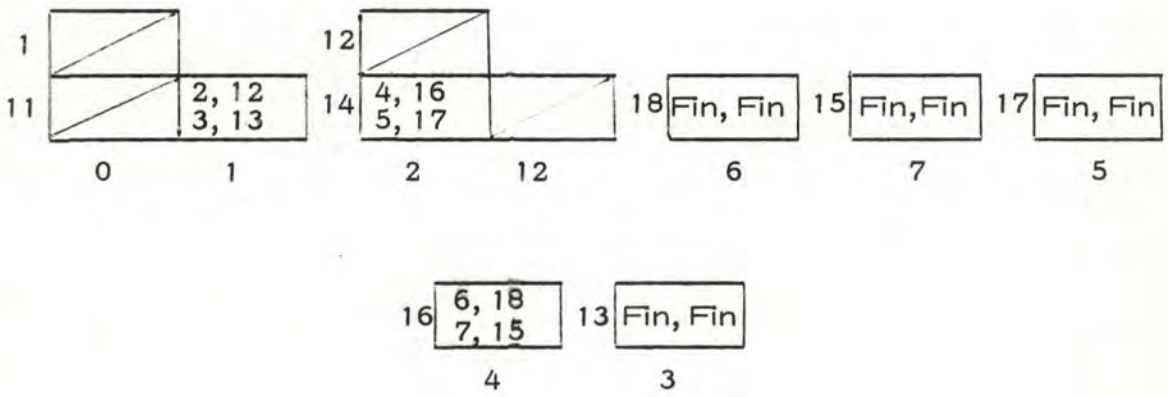
Remarque :

Nous aurions pu établir un tableau complet dans lequel chaque phase de la classe aurait défini une ligne et une

colonne, mais cela aurait été inutile. En effet, la relation d'équivalence, entre les phases est,comme elle le dit par elle-même, une relation d'équivalence au sens mathématique du terme et par conséquent réflexive et symétrique.

b. indiquer dans chaque case du tableau, les couples de phases vers lesquelles mènent soit inconditionnellement, soit conditionnellement par vérification des mêmes conditions d'entrée, les phases qui correspondent à cette case. S'il n'en existe pas, barrer la case(les cases barrées correspondent à des phases non équivalentes).

Exemple :



c. barrer les cases non barrées du tableau qui contiennent un couple de phases dont la case est barrée dans le même tableau.

Exemple : aucune case n'est à barrer.

3. Considérer tous les tableaux globalement; barrer les cases non barrées de chaque tableau qui contiennent un couple de phases dont la case est barrée dans un autre tableau.

Exemple : on est amené à barrer la case (11, 1) par suite de la case barrée (12, 2).

4. Toutes les cases non barrées, après l'étape 3, correspondent aux phases équivalentes.

Exemple : on retrouve les phases équivalentes qui avaient été détectées au paragraphe 2. 5. 1.

Remarque :

Les étapes 2 et 3 de la méthode consistent à tester la deuxième condition qui doit être vérifiée pour qu'il y ait équivalence.

2. 5. 3. Pseudo-équivalence de phases.

Rappelons que dans le corps d'une table condensée peuvent apparaître des tirets, ceux-ci peuvent avoir 3 significations :

1. cas impossible, c'est-à-dire que la condition qui référence la colonne du tiret ne sera jamais vérifiée;
2. cas indifférent c'est-à-dire que si la condition qui référence la colonne du tiret est vérifiée, aucune décision n'est prise;
3. cas d'anomalie c'est-à-dire que la condition qui référence la colonne du tiret ne peut jamais être vérifiée; si elle l'était, ce serait anormal. Dans ce cas, il est conseillé de substituer au tiret une phase d'erreur à laquelle on se branchera si jamais la condition était malgré tout vérifée.

Deux phases enregistrées dans une table condensée, sont pseudo-équivalentes si :

1. elles déterminent le même traitement à exécuter ;
2. elles mènent par vérification des mêmes conditions d'entrée :
   - soit toutes deux à des phases équivalentes, identiques ou pseudo-équivalentes;
   - soit l'une à une phase et l'autre à un cas impossible identifié par un tiret.

Exemple : soit la table condensée :

PHASES	TRAIT.	a	b
1	T_1	3	-
2	T_1	-	4
3	T_2	5	5
4	T_2	6	-
5	T_3	-	1
6	T_3	-	2

Fig. II-6

dans laquelle tous les tirets correspondent à des cas n'apparaissant jamais.
Les phases 1 et 2, 3 et 4, 5 et 6 sont pseudo-équivalentes.

Remarque :

La relation de pseudo-équivalence entre phases est moins restrictive que la relation d'équivalence (deux phases équivalentes sont pseudo-équivalentes) ; contrairement à celle-ci, la relation de pseudo-équivalence n'est pas transitive.

2. 5. 4. Fusion des phases équivalentes et pseudo-équivalentes.

Nous pouvons simplifier les procédures en fusionnant parmi les phases qui les composent, celles qui sont équivalentes. Ceci se représente au niveau de la table par une fusion des lignes associées à ces phases équivalentes. Phases et lignes fusionnées définissent un seul et même état de la procédure.

Parmi les phases restantes, nous pouvons encore, s'il en existe et si c'est possible, fusionner certaines phases pseudo-équivalentes. Il faut cependant agir avec prudence; on ne peut pas affirmer que toutes les phases pseudo-équivalentes peuvent être fusionnées, en ce sens que la pseudo-équivalence est une relation non transitive. Le meilleur moyen de le faire comprendre est de le mettre en évidence par un exemple : .

Soit la partie de table condensée :

PHASES	TRAIT.	a	b	c
1	T_1	-	4	-
2	T_1	5	-	-
3	T_1	6	-	7

les phases 1 et 2, de même que 1 et 3 sont pseudo-équivalentes. Toutefois il est clair que toutes trois ne pourront être fusionnées en une seule phase.

Pour ne pas commettre d'erreur et tenir compte de l'impact de la fusion de phases pseudo-équivalentes sur d'éventuelles fusions ultérieures, il est conseillé d'agir comme suit : chaque fois qu'on décide de fusionner deux phases pseudo-équivalentes, remplacer les tirets, s'il en existe, par les phases requises. Ainsi dans l'exemple précédent, si nous décidons de fusionner 1 et 2, les tirets seront substitués comme suit,

PHASES	TRAIT.	a	b	c
1	T_1	(5)	4	-
2	T_1	5	(4)	-
3	T_1	6	-	7

et il ne sera dès lors plus possible de fusionner 1 et 3;
Si au contraire, nous décidons de fusionner 1 et 3, les tirets deviennent :

PHASES	TRAIT.	a	b	c
1	T_1	(6)	4	(7)
2	T_1	5	-	-
3	T_1	6	(4)	7

Remarques :

- certaines phases sont déjà fusionnées au moment de la mise en page de la procédure;
- simplification et réduction ne sont pas uniques pour les tables condensées puisque la notion de pseudo-équivalence n'est pas transitive.

Donnons, pour clôturer ce paragraphe, les tables obtenues par fusion des phases équivalentes et pseudo-équivalentes dans les tables des figures II-5 et II-6.

PHASES	TRAIT	B. I.	C_1 0	C_1 1	C_2 0	C_2 1	C_3 0	C_3 1	C_4 0	C_4 1
0	-		1	11						
1	-				2	3				
2	T_0						4	5		
3	R	FIN								
4	T_4								6	7
5	T_3	FIN								
6	T_1	FIN								
7	T_2	FIN								
11	-				12	3				
12	T_0								2	7

PHASES	TRAIT.	a	b
1	T_1	3	3
3	T_2	5	5
5	T_3	-	1

2. 6. Eclatement et chaînage des tables séquentielles.

Nous avons jusqu'à présent raisonné sur des tables uniques; dans la réalité, une seule table ne suffira pas toujours à synthétiser une procédure dans sa totalité en ce sens que si la procédure est importante, la table séquentielle dans laquelle elle sera enregistrée prend facilement de l'ampleur. Il serait par conséquent intéressant de pouvoir découper et chaîner les tables afin de les réduire. Ce même problème a été rencontré pour les tables de décision classiques.

Il existe plusieurs façons d'éclater une procédure en plusieurs tables. Nous nous contenterons de donner la philosophie générale de deux d'entre elles.

1. Aller du général au détaillé en réalisant une mise en page modulaire. Ce principe n'est pas caractéristique aux tables; il est conseillé lors de toute mise en page d'un problème important.
   L'idée est de traduire la procédure au moyen de modules globaux, articulés les uns aux autres par des conditions et d'enregistrer ces modules et leur enchaînement dans une table; ensuite recommencer le raisonnement pour chacun des modules définis.
   Nous obtenons ainsi des tables appartenant à des niveaux de plus en plus détaillés.

2. Eclater la procédure et son enregistrement dans plusieurs tables chaînées. Lors du déroulement de la procédure, une table initiale recevra la main; ensuite selon les phases qui se succédent et les conditions qui sont vérifiées, cette table donnera la main à d'autres tables.

   Elle peut donner la main de deux manières :

   - de manière ouverte (symbolisé par GOTO) : la table à laquelle elle passe le contrôle, une fois consultée, donne soit elle-même la main à une autre table, soit termine la procédure;

- de manière fermée (symbolisé par DO/RETURN ou PERFORM) : la table à laquelle elle donne la main, une fois consultée, lui repasse le contrôle.

Exemple : Chapitre V - Application III.

Nous devons réaliser une analyse syntaxique des programmes BASIC. Il est impossible de la mettre en page dans une seule table. Nous allons donc l'éclater en plusieurs tables. Une première table analysera le premier symbole des instructions, en déduira le type de l'instruction et passera alors le contrôle (via un GOTO) à une table chargée de l'analyse des instructions de ce type.
En outre, toutes les tables consacrées à l'analyse des instructions d'un type particulier pourront faire appel (via un DO/RETURN) à la table chargée de l'analyse des expressions.

CHAPITRE III

LA TABLE SEQUENTIELLE : OUTIL D'ANALYSE

3. 1. Introduction : qu'entend-on par analyse ?

Après mise en page d'une procédure sous forme de table séquentielle, nous aimerions à partir de cette table l'analyser.

L'analyse consiste à :

1. Rechercher toutes les boucles de la procédure en spécifiant les phases qui les composent. Une distinction devra être faite entre les boucles normales et anormales. Ces dernières sont celles qui risquent de provoquer un bouclage sans fin de la procédure
   - soit qu'elles sont engendrées par des branchements inconditionnels en cascade. Les phases qui composent les boucles de ce type sont à éliminer d'office ainsi que les phases qui y mènent inconditionnellement;
   - soit que ne se présente jamais une des conditions d'entrée qui permet à la procédure d'en sortir.

2. Déterminer toutes les séquences possibles de la procédure c'est-à-dire toutes les suites de traitements susceptibles d'être exécutés en séquence, en fonction des conditions d'entrée.

3. Souligner les cas non envisagés dans la table. Si la table est condensée, ces cas sont signalés à vue par les cases vides; si elle est détaillée, un cas non envisagé est signalé lorsqu'une ligne ne spécifie pas deux phases suivantes pour la phase qui lui est associée. Il s'agira d'établir toutes les sous-séquences qui aboutissent à ces cas non prévus afin que l'analyste puisse conclure
   - soit d'un cas omis volontairement parce que ne survenant jamais;
   - soit d'un cas oublié et par conséquent à introduire.

Cette analyse de la procédure, à partir de la table, apporte des précisions, et quant aux cas ou séquences prévus, et quant aux cas non prévus. Elle permettra à l'analyste de vérifier d'une part, si le problème a été complètement et correctement défini et d'autre part, si la mise en page traduit bien le problème.

Pour résoudre ces différents problèmes, nous ferons appel à la théorie des graphes en associant à chaque table un graphe.

3. 2. Graphe associé à une table séquentielle.

" Un graphe est un ensemble d'éléments appelés sommets ou noeuds, en nombre fini ou non mais distincts et dénombrables et reliés entre eux par des branches orientées appelées arcs. Un graphe est entièrement défini par un couple (X, U où X) est l'ensemble des sommets et U l'ensemble des arcs".

"Un 1-graphe est un graphe tel que deux sommets quelconques sont l'un extrémité initiale, l'autre extrémité finale d'un arc au plus" (*).

A toute table séquentielle, on peut associer un graphe (plus exactement un 1-graphe) en définissant les ensembles X et U de la manière suivante :

- chaque phase enregistrée dans la table définit un sommet ou noeud du graphe;
- soit i et j, 2 sommets : un arc est défini de i vers j, si et seulement si la phase associée au sommet i admet la phase associée au sommet j comme phase suivante pour au moins une condition d'entrée.

(Dans la suite, nous parlerons de graphe pour désigner ce 1-graphe).

Exemple : Soient les tables séquentielles :

PHASES	TRAIT.	B. I.	a	b	c
1	T0		2	2	3
2	T1	5	-	-	-
3	T2		4	-	4
4	T1		-	5	-
5	T3		1	-	6
6		FIN	-	-	-

Fig. III-1 a.

(*) Notes du cours : "Théorie des graphes" - J. Fichefet - Namur.

PHASES	TRAIT.	B. I.	C_1		C_2		C_3	
			0	1	0	1	0	1
1	T0		2	3				
2	T1				4	5		
3	T2				6	5		
4	T1	7						
5	-						6	7
6	-						6	8
7	T3	1						
8	T4	FIN						

Fig. III-2a

et les graphes qu'elles définissent :

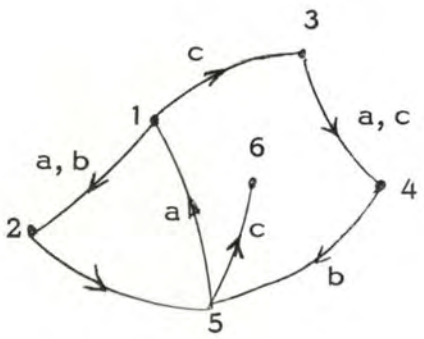


Fig. III-1 b

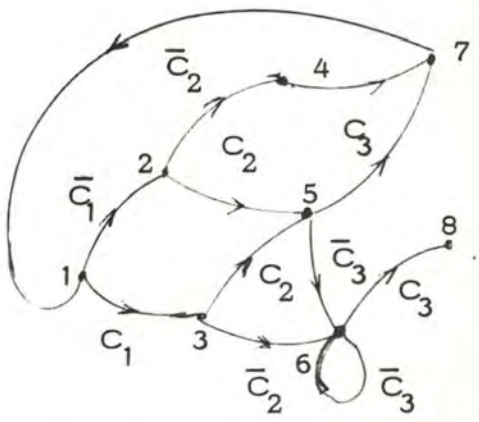


Fig. III-2 b

Ce graphe est un autre moyen d'enregistrer la succession des phases. Nous pouvons l'enrichir en surmontant les arcs de la (des) condition(s) qui détermine(nt) les branchements entre les phases associées aux différents sommets. Si le branchement d'une phase à une autre est inconditionnel, l'arc qui le caractérise ne sera surmonté d'aucune indication.

La succession des phases peut aussi être rendue par la matrice booléenne associée au graphe, encore appelée matrice de transition. Pour l'exemple de la figure III-1 :

$$M = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

avec M (i, j) = 1 s'il existe un arc qui relie le sommet i au sommet j.

= 0 sinon.

Puisqu'à toute table séquentielle, nous pouvons associer un graphe, nous travaillerons sur ce graphe pour résoudre les différents problèmes posés : en effet

1. Rechercher les boucles de la procédure revient à chercher les circuits du graphe;

2. Déterminer toutes les séquences possibles de la procédure revient à chercher tous les chemins possibles entre le sommet associé à la phase initiale et les sommets associés aux phases finales.

3.3. Détermination des boucles dues aux branchements inconditionnels. Détermination des phases qui y mènent inconditionnellement.

Il s'agit de détecter le premier type de boucles anormales c'est-à-dire celles qui provoquent un bouclage sans fin de la procédure. Ces boucles (les phases qui les constituent) sont à éliminer. Ensuite, il convient aussi d'éliminer les phases qui inconditionnellement mènent à ces boucles.

3. 3. 1. Sous-graphe inconditionnel.

Nous n'allons pas travailler sur le graphe associé à la table lui-même mais sur le sous-graphe obtenu en ne considérant que les arcs inconditionnels et les sommets qui en sont les extrémités. Le problème consiste donc à déterminer les circuits éventuels de ce sous-graphe et les sommets qui sont ascendants des sommets qui y appartiennent (un sommet est ascendant d'un autre s'il existe un chemin partant de ce sommet et aboutissant à l'autre).

Exemple : soit le sous-graphe inconditionnel :

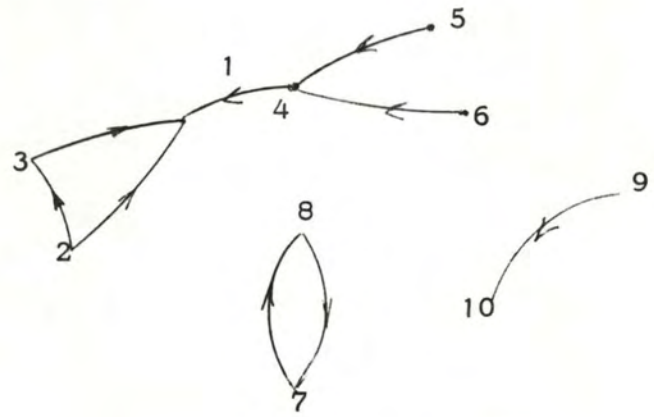


Fig. III-3

et sa matrice :

$$M = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \end{array} \begin{array}{c} \begin{array}{cccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{array} \\ \left(\begin{array}{cccccccccc} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \right) \end{array}$$

Ce sous-graphe et sa matrice possèdent la caractéristique suivante : chaque sommet est extrémité initiale d'un arc au plus ; chaque ligne de la matrice comprend au plus un "1".

3. 3. 2. Méthode utilisée.

Nous nous baserons sur la propriété des matrices associées aux graphes qui dit que "le coefficient (i, j) de M^k indique le nombre de chemins de longueur k partant de i et aboutissant en j. En particulier, si i = j, M^k (i, i) représente le nombre de circuits de longueur k auxquels appartient le sommet i" (*).

En vertu de la caractéristique du sous-graphe, M^k ne comportera que des "1" ou des "0" (ce qui traduit le fait qu'il existe au plus un chemin de longueur k entre 2 sommets du sous-graphe) et au plus un "1" par ligne (ce qui traduit le fait que de tout sommet part au plus un chemin de longueur k).

La procédure de recherche consiste à calculer en partant de M, les matrices successives M^2, M^3, ... M^n, si le graphe possède n sommets (nous verrons plus loin pourquoi il n'est pas nécessaire d'aller au-delà de l'ordre n).

Les circuits se détectent par examen des diagonales de ces matrices, les sommets ascendants des sommets qui appartiennent aux circuits, par examen des colonnes correspondant, et aux sommets des circuits et aux sommets déjà reconnus ascendants de ces sommets :

soit M^k $\quad k \in \{1, 2, 3 \ldots n\}$

1. $\forall i$: M^k (i, i) = 1 implique que le sommet i appartient à un circuit (de longueur k).
   D'une part, la phase associée à ce sommet est dangereuse et à éliminer de la procédure. D'autre part, ce sommet sera ignoré dans la suite de la recherche puisqu'il ne peut, en vertu des caractéristiques du sous-graphe, appartenir à un autre circuit de longueur $>$ k, qui contienne des sommets différents de ceux découverts ou mener à un autre circuit. Autrement dit, la suppression de la ligne et de la colonne de ce sommet dans les matrices M et M^k ne nous empêchera pas de découvrir tous les sommets désirés.

2. Soit un sommet i, appartenant à un circuit de longueur k (par application de 1) ou ascendant d'un sommet appartenant à un tel circuit (déjà détecté par application de 2)); $\forall j$: M^k (i, j) = 1 implique que j est sommet ascendant d'un sommet appartenant à un circuit de longueur k. Ce sommet correspond à une phase qui doit également être éliminée comme menant inconditionnellement à une boucle inconditionnelle. Lui-même peut être ignoré pour la suite de la

---

(*) Notes de cours "Théorie des graphes" - J. Fichefet - Namur.

recherche par suppression de sa ligne et de sa colonne dans M et M^k, en effet, étant donné un circuit de longueur k, par application répétée de 2, nous obtenons <u>tous</u> les sommets ascendants des sommets de ce circuit.

La procédure de recherche se termine lorsque la matrice M^k est soit nulle, soit de dimension nulle. Ceci surviendra au plus tard en M^{n+1}. En effet, nous ne pouvons pas découvrir des circuits de longueur $n + i$ ($i \geq 1$), même s'il en existe réellement dans le sous-graphe, car les sommets qui les constituent appartiennent, en vertu des caractéristiques du sous-graphe, à des circuits de longueur $\leq n$ et auront donc été supprimés; nous ne découvrirons pas davantage de chemins de longueur $n + i$ ($i \geq 1$) car en vertu du fait qu'il n'y a que n sommets dans le sous-graphe, et de ses caractéristiques, les sommets de ce chemin mènent à un circuit et auront été supprimés.

3. 3. 3. <u>Exemple.</u>

Reprenons le sous-graphe de la figure III. 3.

<u>Circuit de longueur 1</u> : la diagonale de M est nulle, il n'en existe pas;

<u>Circuit de longueur 2</u> : calculons $M^2 = \underbrace{M \times M}$

produit matriciel ligne par colonne

$$M^2 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \end{array} \begin{array}{c} \begin{array}{cccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{array} \\ \left(\begin{array}{cccccccccc} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \right) \end{array}$$

- les sommets 7 et 8 appartiennent à un circuit de longueur 2
- les sommets 7 et 8 n'admettent pas d'ascendants : en effet, les colonnes 7 et 8 sont nulles en dehors de la diagonale

- les lignes et colonnes 7 et 8 peuvent être supprimés; soient M' et M'^2 les matrices ainsi obtenues.

Circuit de longueur 3 : calculons $M^3 = M'^2 \times M'$

$M^3 =$

	1	2	3	4	5	6	9	10
1	1	0	0	0	0	0	0	0
2	0	1	0	0	0	0	0	0
3	0	0	1	0	0	0	0	0
4	0	0	1	0	0	0	0	0
5	0	1	0	0	0	0	0	0
6	0	1	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0

- les sommets 1, 2, 3 appartiennent à un circuit de longueur 3;
- les sommets 5, 6, 4 sont des sommets ascendants de sommets appartenant à un circuit de longueur 3 : en effet, les colonnes 2, 3 comportent un "1" en ligne 4, 5, 6;
- avant de poursuivre, supprimons les lignes et colonnes 1, 2, 3, 4, 5, 6. Soient les matrices M' et M'^3 ainsi obtenues.

Circuit de longueur 4 : la matrice obtenue M'^3 est nulle; il n'ya plus de ciruits à découvrir.

3. 3. 4. Conclusion.

Toutes les phases associées aux sommets qui appartiennent à un circuit ou qui sont ascendants de tels sommets sont à éliminer de la table et de la procédure.

De telles boucles peuvent provenir soit d'une erreur de logique au niveau de la conception du problème, soit d'une faute d'attention au niveau de l'enregistrement de la succession des phases dans la table séquentielle.

3. 4. Détermination des boucles et séquences possibles de la procédure.

Après avoir élagué la table séquentielle, le graphe et sa matrice, des phases et sommets qui déterminent des bouclages sans fin, ou qui y mènent inconditionnellement, nous allons déterminer les suites des phases qui peuvent se succéder entre la phase initiale et les différentes phases finales, ainsi que les boucles éventuelles de la procédure.

En travaillant sur le graphe associé à la table, le problème revient à déterminer tous les chemins qui existent entre le sommet

initial (associé à la phase initiale) et les sommets terminaux (associés aux phases finales), ainsi que les circuits du graphe. Plus précisément, il suffit de déterminer tous les chemins élémentaires entre les sommets précités et les circuits élémentaires (entendons par là les chemins et circuits qui n'empruntent pas deux fois le même sommet exception faite dans le cas des circuits, pour leur sommet initial et final qui coïncident). En effet, à partir de ces derniers, il est possible de reconstituer tous les chemins et circuits du graphe.

3. 4. 1. Méthode de la multiplication latine (*).

"Tout chemin dans un graphe peut être représenté par la séquence des lettres ou chiffres qui identifient ses sommets. Si ce chemin est élémentaire, la séquence correspondante est sans répétition, on dit qu'elle est "latine"

Grâce à une multiplication matricielle d'un type particulier, il est possible de déterminer successivement tous les chemins élémentaires de longueur 1, 2 ... n-1 et les circuits élémentaires de longueur 1, 2, ... n (s'il y a n sommets dans le graphe).

La matrice latine $M^{(r)}$ à n lignes et à n colonnes jouit de la propriété suivante : les éléments $M^{(r)}$ (i, j) représentent l'ensemble des chemins élémentaires de longueur r partant du sommet i et aboutissant au sommet j; en particulier si i = j, $M^{(r)}$ (i, i) représente l'ensemble des circuits élémentaires de longueur r auxquels appartient le sommet i.
Si $M^{(r)}$ (i, j) est vide, il n'existe pas de tel chemin ou circuit.

Construction des matrices latines $M^{(r)}$:

1. Soit $M^{(1)}$: s'il existe un arc de i à j dans le graphe, nous portons la séquence i, j dans la case (i, j) de $M^{(1)}$; nous la laissons vide dans le cas contraire;

2. De $M^{(1)}$, nous déduisons $\overline{M}$ en supprimant le premier élément de chaque séquence dans $M^{(1)}$ (si elle existe);

3. $M^{(2)}$: pour l'obtenir nous multiplions $M^{(1)}$ par $\overline{M}$ ligne par colonne d'une manière particulière:
lorsqu'une case (i, j) de $M^{(1)}$ coïncide avec une cas (j, k) de M, nous laissons vide la case (i, k) de $M^{(2)}$, si pour tous les j, l'une et/ou l'autre de ces cases est vide ou bien si pour tous les j les cases sont telles que nous n'obtenons pas de séquence latine en concaténant la (les) séquence(s) de la case (i, j) de $M^{(1)}$ avec la (les) séquence(s) de la case (j, k) de $\overline{M}$.

(*) Méthodes et modèles de la recherche opérationnelle (Tome II) - Dunod 1964 - (p. 307-315) IN : Notes de cours :"Théorie des graphes" - J. Fichefet - Namur.

Si en concaténant les séquences (i, j) de $M^{(1)}$ avec les séquences (j, k) de $\overline{M}$, pour les différents j, nous trouvons une ou plusieurs séquences latines, nous les portons toutes dans la case (i, k) de $M^{(2)}$.
Il n'est pas difficile de s'apercevoir que $M^{(2)}$ donne tous les chemins et circuits élémentaires de longueur 2.

4. Pour obtenir les matrices latines de $M^{(3)}$, $M^{(4)}$ ... $M^{(n-1)}$ on procède de même :

$$M^{(3)} = M^{(2)} \times \overline{M}$$

$$M^{(4)} = M^{(3)} \times \overline{M}$$

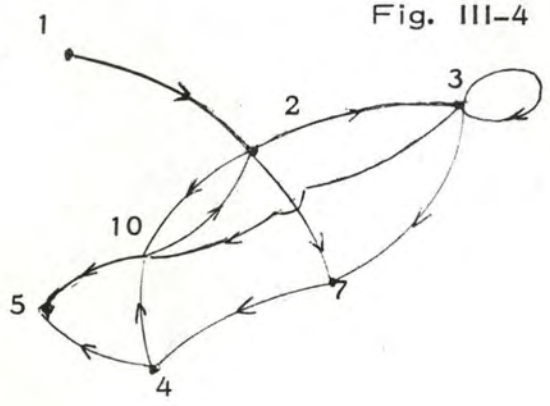
$$M^{(n-1)} = M^{(n-2)} \times \overline{M}$$

Puisque nous ne nous intéressons qu'aux circuits et aux chemins partant du sommet initial et aboutissant aux sommets terminaux, dans chaque matrice latine $M^{(r)}$, nous n'examinerons que les séquences qui figurent dans les cases diagonales et dans les cases (1, i) pour tout i sommet terminal (en supposant le sommet 1 sommet initial).

Exemple : soit la table séquentielle et son graphe associé:

PHASES	TRAIT.	B. I.	C1	C2	C3
1	T0		2	-	2
2	T1		10	7	3
3	T2		10	7	3
4	T1		10	5	10
5	T3	FIN	-	-	-
7	T1		4	-	4
10	T4		5	2	2

Fig. III-4



phase initiale = 1
phase finale = 5

Chemins et circuits élémentaires de longueur = 1.

$M^{(1)} =$

	1	2	3	4	5	7	10
1		1, 2					
2			2, 3			2, 7	2, 10
3			3, 3			3, 7	3, 10
4					4, 5		4, 10
5							
7				7, 4			
10		10, 2			10, 5		

- par examen de la case (3, 3), nous concluons d'une boucle par 3 ;
- la case (1, 5) est vide : pas de chemin de longueur 1.

$\overline{M} =$

	1	2	3	4	5	7	10
1		2					
2			3			7	10
3			3			7	10
4					5		10
5							
7				4			
10		2			5		

Chemins et circuits élémentaires de longueur = 2.

M^2 =

	1	2	3	4	5	7	10
1			1, 2, 3			1, 2, 7	1, 2, 10
2		2, 10, 2		2, 7, 4	2, 10, 5	2, 3, 7	2, 3, 10
3		3, 10, 2		3, 7, 4	3, 10, 5		
4							
5							
7					7, 4, 5		7, 4, 10
10			10, 2, 3			10, 2, 7	10, 2, 10

- la case (1, 5) est vide;
- la case (2, 2) ou (10, 10) nous donne un circuit de longueur 2 : 2, 10, 2.

Chemins et circuits élémentaires de longueur = 3.

M^3 =

	1	2	3	4	5	7	10
1				1, 2, 7, 4	1, 2, 10, 5	1, 2, 3, 7	1, 2, 3, 10
2		2, 3, 10, 2		2, 3, 7, 4	2, 3, 10, 5 2, 7, 4, 5 2, 3, 10, 5		2, 7, 4, 10
3			3, 10, 2, 3		3, 7, 4, 5	3, 10, 2, 7	3, 7, 4, 10
4			4, 10, 2, 3			4, 10, 2, 7	
5							
7		7, 4, 10, 2			7, 4, 10, 5		
10				10, 2, 7, 4		10, 2, 3, 7	10, 2, 3, 10

- cette fois la case (1, 5) nous donne le chemin élémentaire 1, 2, 10, 5;
- les cases diagonales le circuit : 3, 10, 2, 3.

Chemins et circuits de longueur = 4, 5, 6.

En procédant comme précédemment, on découvre par examen des cases (1, 5) et diagonales des matrices $M^{(4)}$, $M^{(5)}$, $M^{(6)}$ les chemins

- 1, 2, 7, 4, 5
- 1, 2, 3, 10, 5
- 1, 2, 3, 7, 4, 5
- 1, 2, 7, 4, 10, 5
- 1, 2, 3, 7, 4, 10, 5

et les circuits

- 10, 2, 7, 4, 10
- 2, 3, 7, 4, 10, 2

Les cases diagonales de $M^{(7)}$ sont nulles.

Conclusion.

La méthode de la multiplication latine a permis d'obtenir tous les circuits élémentaires et tous les chemins élémentaires; ces derniers de manière non redondante.

Cette méthode, facilement programmable, présente cependant un inconvénient : les matrices latines occupent, beaucoup de place. Comme nous désirons implémenter la procédure de recherche des chemins et boucles, nous allons, pour éviter cet inconvénient, mettre au point une autre méthode.

3. 4. 2. Méthode pas à pas.

Cette méthode requiert beaucoup moins de place que la méthode de la multiplication latine. Elle permet d'énumérer tous les circuits et sans redondance, tous les chemins élémentaires du graphe associé à la table, entre sommet initial et sommets terminaux, ce qui en vertu de la bijection qui a été établie entre les sommets du graphe et les phases revient à énumérer toutes les boucles et chemins élémentaires de la procédure entre phase initiale et phases finales.

L'idée est de partir du sommet initial pour procéder ensuite par tâtonnements, en envisageant pour chaque sommet tous ses sommets successeurs (un sommet i est successeur d'un sommet j, s'il est extrémité finale d'un arc admettant j pour extrémité initiale).

Nous utiliserons pour rendre compte de la succession des sommets, un stack LIFO. Ce stack, puisque nous ne nous intéressons qu'aux circuits et chemins élémentaires, aura un nombre maximum d'entrée égal à n (s'il y a n phases enregistrées dans la table).

L'algorithme peut s'énoncer comme suit :

1. Placer le sommet initial dans le stack (initialement vide).
   Aller en 2.

2. Le dernier sommet introduit dans le stack admet-il un successeur qui n'a pas encore été pris en considération ?
   - si oui : l'introduire au sommet du stack.
     Aller en 3.
   - si non ; supprimer la dernière entrée du stack.
     Aller en 2.

3. Le dernier sommet introduit dans le stack est-il sommet terminal ?
   - si oui : la suite des sommets enregistrée dans le stack constitue un chemin élémentaire entre le sommet initial et un sommet final.
     Supprimer la dernière entrée.
     Aller en 2.
   - si non : Aller en 4.

4. Le dernier sommet introduit dans le stack y figure-t-il déjà ?
   - si oui : nous avons un circuit élémentaire composé des sommets enregistrés dans le stack entre les deux occurrences du même sommet, et de ce dernier.
     Supprimer la dernière entrée.
     Aller en 2.
   - si non : Aller en 2.

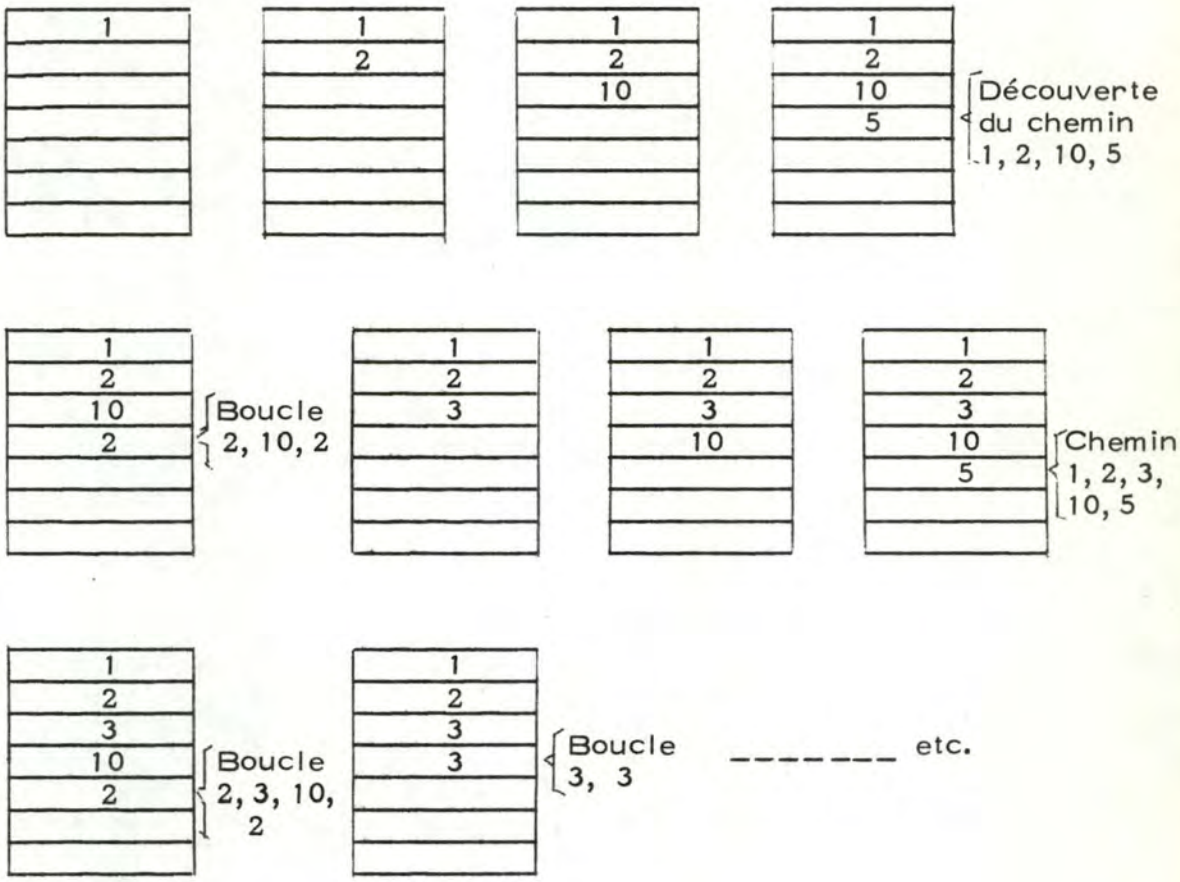
Remarquons que :

- cet algorithme se termine en un nombre fini d'itérations;
- il fournit sans redondance, tous les chemins élémentaires entre sommet initial et sommets terminaux.

D'autre part, les chemins et les circuits sont sortis dans l'ordre de leur construction.

Exemple :

Reprenons la table séquentielle de la figure III-4.
Les contenus successifs du stack sont les suivants :



Nous obtenons les mêmes chemins et circuits élémentaires que par la méthode précédente et cette fois sans utiliser l'outil matriciel. Le seul élément requérant de la place est le stack.

Remarque :

Cet algorithme, nous avons découvert, qu'il était en beaucoup de points semblable à l'algorithme de Tiernan (algorithme qui permet de déterminer tous les circuits élémentaires d'un graphe (1-grpahe)) (*).

Le procédé de construction des circuits ou chemins est identique : essayer d'allonger tout chemin élémentaire obtenu en considérant tous les successeurs du dernier sommet constituant ce chemin.

(*) J. C. Tiernan "An efficient search algorithm to find the elementary circuits of a graph "CACM-vol 13 - 1970 (p. 722-726) IN Notes de cours "Théorie des graphes" - J. Fichefet- Namur.

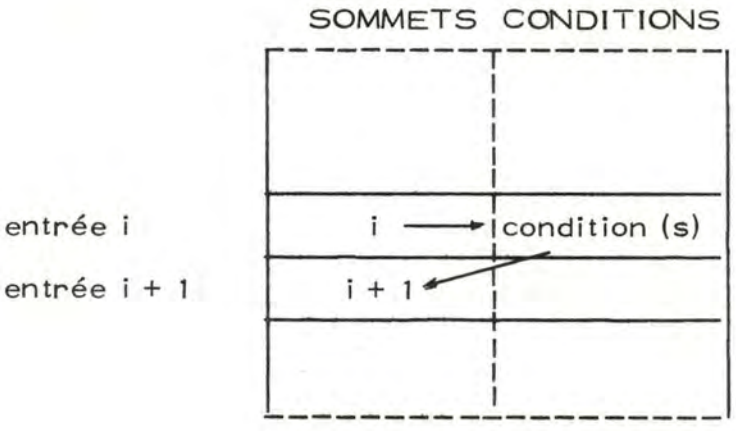
Des différences :

- l'algorithme pas à pas fournit tous les chemins et circuits élémentaires tandis que l'algorithme de Tiernan fournit tous les circuits élémentaires et de la manière dont il procède, certains chemins élémentaires;
- l'algorithme pas à pas ne considère que les chemins élémentaires entre certains sommets (le sommet initial et les sommets terminaux).

3. 4. 3. Extension de la méthode pas à pas.

Par les méthodes que nous avons exposées, nous obtenons les séquences et boucles de la procédure, caractérisées par les phases qui les composent. Pour faciliter la tâche de l'analyste qui doit tester si la procédure a été correctement définie et mise en page, il y a intérêt à donner les séquences et les boucles en spécifiant les traitements qui les composent et les conditions qui les articulent.

Pour ce faire, il suffit dans la méthode pas à pas d'enrichir chaque entrée du stack d'une information supplémentaire : la ou lesconditions dont la vérification provoque le branchement de la phase associée au sommet correspondant à cette entrée vers la phase associée au sommet correspondant à l'entrée suivante dans le stack. Le stack devient donc double :



Remarque :

Il existe deux cas particuliers où aucune condition ne figurera :

- le branchement est inconditionnel;
- le sommet est sommet terminal.

Reprenons l'exemple précédent, les contenus des deux stacks au moment de la découverte du 1er chemin et de la 1ère boucle, par exemple, sont :

SOMMETS	CONDITIONS
1	C1 , C3
2	C1
10	C1
5	

SOMMETS	CONDITIONS
1	C1 , C3
2	C1
10	C2 , C3
2	

Cette façon de procéder permettra, lors de la découverte d'un chemin ou d'une boucle, de spécifier les conditions qui articulent les branchements entre phases qui les composent; quant aux traitements, il suffit pour chaque phase de donner le traitement qui lui est associé c'est-à-dire le traitement qui est exécuté lorsqu'elle a la main.

Le 1er chemin déterminé dans l'exemple correspond à la séquence :

PHASES	COND/TRAIT.
1	T0
	C1 ou C3
2	T1
	C1
10	T4
	C1
5	T3

Cette méthode pas à pas pourrait en fait manipuler directement les phases plutôt que les sommets qui leur sont associés dans le graphe. Il suffirait de remplacer dans l'algorithme les termes "sommet" et "successeur" respectivement par "phase" et "suivante". Mais, comme nous allons implémenter cet algorithme (voir paragraphe 3. 6.), il est plus intéressant de travailler à partir du graphe; en effet, dans la table, pour deux conditions d'entrée différentes, on peut avoir la même phase suivante alors que dans le graphe, cela correspond à un seul sommet successeur. En travaillant à partir du graphe plutôt qu'à partir de la table on évitera donc des tests d'égalité, puisqu'ils auront été effectués une fois au moment

de l'élaboration du graphe à partir de la table tandis qu'en travaillant à partir de la table, ils doivent être effectués sans cesse au cours de l'algorithme lui-même

3. 5. Détermination des sous-séquences aboutissant aux cas omis.

Jusqu'à présent l'analyse s'est limitée à l'étude des séquences enregistrées dans la table. Elle doit également prélever et donner des précisions au sujet des cas non prévus. Ce sera l'objet de ce paragraphe.

Un moyen simple est de détecter les cas omis au cours de la recherche des chemins et boucles de la procédure, plutôt que de définir une méthode propre à la recherche de ces cas.

Il suffit donc de reprendre la méthode pas à pas et d'y insérer une étape supplémentaire définie ainsi :

lorsqu'il n'existe plus (étape 2), pour le dernier sommet introduit dans le stack, de sommet successeur non déjà considéré, on se posera la question :

- existe-t-il une (ou plusieurs) condition(s) d'entrée pour laquelle la phase associée au dernier sommet du stack n'admet pas de phase suivante ?
  (si la table est condensée)
  - si non : poursuivre l'algorithme normalement;
  - si oui : avant de poursuivre l'algorithme, signaler que les sommets enregistrés dans le stack à ce moment-là, constituent une sous-séquence aboutissant à un ou plusieurs cas omis.

- le dernier sommet introduit dans le stack a-t-il admis deux sommets successeurs (correspondant l'un à la vérification, l'autre à la non vérification, d'une condition d'entrée ?
  (si la table est détaillée)
  - si oui : poursuivre l'algorithme normalement;
  - si non : avant de poursuivre l'algorithme normalement signaler que les sommets enregistrés dans le stack à ce moment-là, constituent une sous-séquence aboutissant à un ou plusieurs cas omis.

3. 6. Analyse automatique.

Nous allons implémenter la procédure de recherche des boucles inconditionnelles et des phases qui y mènent inconditionnellement (partiel) ainsi que la méthode pas à pas de détermination des séquences et boucles enregistrées dans la table, incluant la recherche des sous-séquences aboutissant aux cas omis (partie II).

Comme données nous avons :

- la table séquentielle (T);
- le vecteur des phases (E);
- le vecteur des conditions (C$);
- le vecteur des traitements (T$);
- n le nombre de phases enregistrées dans la table ;
- c le nombre de conditions (y compris B. I.).

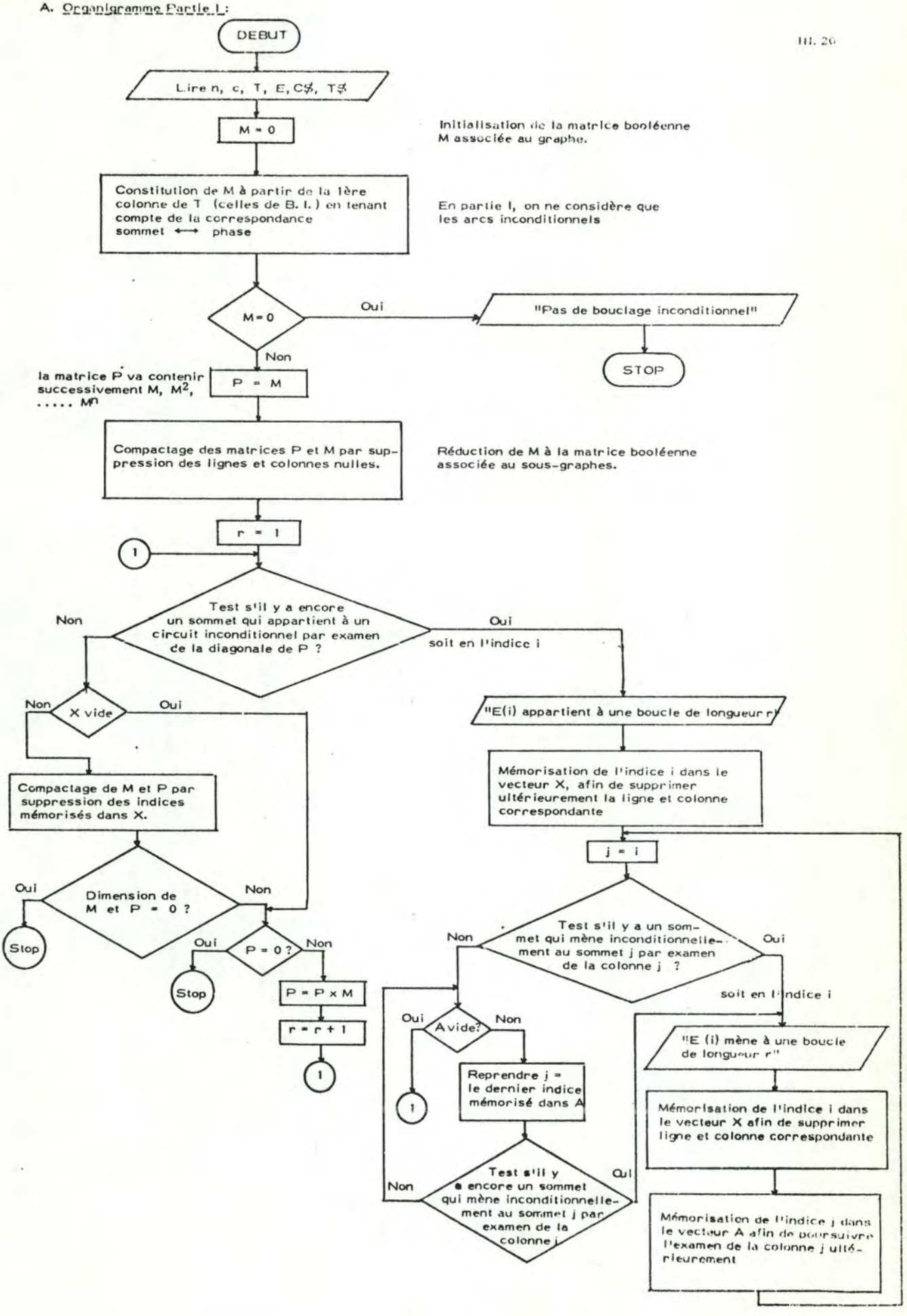
Remarques :

1. Si la table n'est pas trop importante, elle peut être introduite en mémoire centrale, sinon enregistrée ligne par ligne sur un fichier séquentiel disque;

2. Nous avons implémenté l'analyse des tables séquentielles sous forme condensée (forme la plus intéressante et la plus courante de la table). Pour les tables détaillées il suffit, soit de les ramener à la forme condensée (voir chapitre 2), soit de modifier quelque peu le programme;

3. Nous avons choisi le langage de programmation BASIC non pour ses performances (!) mais parce qu'il permet de manipuler sans problèmes des données alphanumériques.

A. Organigramme Partie I :



B. Organigramme partie II.

Il suffit de présenter l'algorithme de la méthode pas à pas sous forme d'organigramme, ce qui ne présente aucune difficulté.
L'algorithme utilise :

- le stack X de dimension n;
- une matrice M de dimension (n, n), qui représente le dictionnaire du graphe associé à la table (ce dernier spécifie pour chaque sommet, quels sont ses sommets successeurs);
- une matrice P, de dimension (n, n), image de M qui permet de tenir à jour les sommets successeurs d'un sommet qui ont déjà été considérés.

C. Programme et exemples.

```
10 DIM T(6,4),M(7,7),P( ,7),Q(7,7),A(7),B(7),E(7)
11 DIM X(7),U(7),F(7),T$(7),C$(4)
12 REM T=TABLE SEQUENTIELLE,E=VECTEUR DES PHASES,T$=VECTEUR DES TRAIT.
13 REM C$=VECTEUR DES CONDITIONS,N=NOMBRE DE PHASES,C=NOMBRE DE COND.
14 REM M=MATRICE BOOL. DU GRAPHE ASSOCIE A LA TABLE
20 REM LECTURE DES DONNEES
30 READ N,C
31 FOR I=1 TO N-1
32 FOR J=1 TO C
33 READ T(I,J)
34 NEXT J
35 NEXT I
36 FOR I=1 TO N
37 READ E(I)
40 NEXT I
41 FOR I=1 TO N
42 READ T$(I)
43 NEXT I
44 FOR I=1 TO C
45 READ C$(I)
46 NEXT I
50 REM IMPRESSION DE LA TABLE
60 PRINT TAB(8),"ANALYSE DE LA TABLE SUIVANTE : "
70 PRINT
80 PRINT
90 PRINT USING 3000
100 PRINT USING 3010,C$(1),C$(2),C$(3),C$(4)
110 FOR I=1 TO 6
120 PRINT USING 3020,E(I),T$(I),T(I,1),T(I,2),T(I,3),T(I,4)
130 NEXT I
140 PRINT USING 3030,E(7),T$(7)
180 FOR I=1 TO N
181 F(I)=E(I)
182 NEXT I
190 REM PARTIE 1
195 MAT M=ZER
200 REM CONSTITUTION DE LA MATRICE DU GRAPHE INCONDITIONNEL
210 FOR I=1 TO N-1
220 IF T(I,1)=0 THEN 270
230 FOR J=1 TO N
240 IF T(I,1)=E(J) THEN 260
250 NEXT J
260 M(I,J)=1
270 NEXT I
280 FOR I=1 TO N
281 FOR J=1 TO N
282 IF M(I,J)<>0 THEN 290
283 NEXT J
284 NEXT I
285 GOTO 1190
290 MAT P=M
300 REM SUPPRESSION DES COLONNES NULLES
310 S=1
320 FOR I=1 TO N
330 FOR J=1 TO N
340 IF M(I,J)=1 THEN 410
350 NEXT J
360 FOR K=1 TO N
370 IF M(K,I)=1 THEN 410
380 NEXT K
390 X(S)=I
400 S=S+1
410 NEXT I
420 N1=N
430 REM N1 DIM. ACT. DES MATRICES
440 REM COMPACT. DE LA MATRICE
```

```
450 GOSUB 1210
460 PRINT
470 PRINT
480 PRINT "RECHERCHE DES PHASES QUI DETERMINENT DES BOUCLES"
481 PRINT " INCONDITIONNELLLES ETVDES PHASES QUI Y CONDUISENT"
482 PRINT " INCONDITIONNELLEMENT"
490 PRINT
500 D1=1
505 REM R EST LA LONGUEUR DES BOUCLES ET DES CHEMINS
510 R=1
520 S=1
530 REM TEST S7IL Y A DES BOUCLES
540 I=1
550 IF P(I,I)=1 THEN 590
560 IF I=N1 THEN 930
570 I=I+1
580 GOTO 550
590 PRINT "LA PHASE";E(I);"APPARTIENT A UNE BOUCLE"
591 PRINT " INCONDITIONNELLE DE LONGUEUR";R
595 U(  )=E(I)
597 D1=D1+1
620 Z=I
630 X(S)=I
640 S=S+1
660 K=1
670 J=I
680 I=1
690 IF P(I,J)=1 THEN 730
700 IF I=N1 THEN 830
710 I=I+1
720 GOTO 690
730 IF I=J THEN 700
740 PRINT "LA PHASE ";E(I);"MENE INC. A UNE BOUCLE"
741 PRINT " DE LONGUEUR";R
745 U( 1)=E(I)
747 D1=D1+1
770 X(S)=I
780 S= +1
790 A(K)=J
800 B(K)=I
810 K=K+1
820 GOTO 670
830 IF K=1 THEN 900
840 J=A(K)
850 I=B(K)
860 K=K-1
870 IF I=N1 THEN 830
880 I=I+1
890 GOTO 690
900 I=Z
910 GOTO 560
930 GOSUB 1210
940 IF N1=0 THEN 1080
950 FOR I=1 TO N1
951 FOR J=1 TO N1
952 IF P(I,J)<>0 THEN 960
953 NEXT J
954 NEXT I
955 GOTO 1080
956 REM MULTIPLICATION DE P PAR M
960 V=0
970 FOR I=1 TO N1
```

```
 980 FOR J=1 TO N1
 985 V=Ø
 990 FOR K=1 TO N1
1000 V=V+P(I,K)*M(K,J)
1010 NEXT K
1020 Q(I,J)=V
1030 NEXT J
1040 NEXT I
1050 MAT P=Q
1060 R= +1
1070 GOTO 520
1080 REM SUPPRESSION DES PHASES SIGNALEES DANS LA TABLE
1090 FOR I=1 TO N-1
1100 FOR J=1 TO C
1110 FOR K=1 TO D1-1
1120 IF T(I,J)=U(K) THEN 1150
1130 NEXT K
1140 GOTO 1160
1150 T(I,J)=Ø
1160 NEXT J
1170 NEXT I
1180 FOR I=1 TO N
1181 E(I)=F(I)
1182 NEXT I
1185 GOTO 1500
1190 PRINT "PAS DE BOUCLAGE INCONDITIONNEL"
1200 GOTO 1500
1205 REM SOUS-ROUTINE DE COMPACTAGE DES MATRICES
1210 IF S=1 THEN 1420
1211 IF S=2 THEN 1224
1212 K=1
1213 I=K
1214 FOR J=K+1 TO S-1
1215 IF X(J)>X(I) THEN 1217
1216 I=J
1217 NEXT J
1218 IF I=K THEN 1222
1219 W=X(K)
1220 X(K)=X(I)
1221 X(I)=W
1222 K=K+1
1223 IF K<>S-1 THEN 1213
1224 K=S-1
1230 I=X(K)
1240 IF I=N1 THEN 1380
1245 IF N1=Ø THEN 1420
1250 FOR J=I TO N1-1
1260 E(J)=E(J+1)
1270 FOR D=1 TO N1
1280 M(J,D)=M(J+1,D)
1290 P(J,D)=P(J+1,D)
1300 NEXT D
1310 NEXT J
1320 FOR J=I TO N1-1
1330 FOR D=1 TO N1-1
1340 M(D,J)=M(D,J+1)
1350 P(D,J)=P(D,J+1)
1360 NEXT D
1370 NEXT J
1380 N1=N1-1
1390 IF K=1 THEN 1420
1400 K=K-1
1410 GOTO 1230
```

```
1420 RETURN
1490 REM PARTIE 2
1500 PRINT
1501 PRINT "LES PHASES PRELEVEES SONT DANGEREUSES ET A ELIMINER"
1502 MAT M=ZER
1510 MAT P=ZER
1550 REM CONSTITUTION DU GRAPHE ASSOCIE A LA TABLE
1570 J1=1
1580 K1=1
1590 FOR J=1 TO N-1
1600 FOR K=1 TO C
1610 IF T(J,K)=0 THEN 1710
1620 IF K=1 THEN 1660
1630 FOR I=1 TO K-1
1640 IF T(J,K)=T(J,I) THEN 1710
1650 NEXT I
1660 FOR Z=1 TO N
1670 IF E(Z)=T(J,K) THEN 1690
1680 NEXT Z
1685 STOP
1690 M(J1,K1)=Z
1700 K1=K1+1
1710 NEXT K
1720 J1=J1+1
1730 K1=1
1740 NEXT J
1745 PRINT
1747 PRINT
1750 PRINT "RECHERCHE DE TOUS LES CHEMINS ET BOUCLES ENTRE"
1751 PRINT " LA PHASE INITIALE ET LES PHASES FINALES"
1752 PRINT "AINSI QUE LES SOUS- EQUENCES ABOUTISSANT AUX CAS OMIS"
1760 PRINT
1765 REM INITIALISATION DU STACK
1770 X(1)=1
1780 I=1
1790 J=1
1795 REM LE NOEUD AU SOMMET ADMET-IL ENCORE UN NOEUD NON CONSIDERE
1800 IF P(X(I),J)=1 THEN 2140
1810 P(X(I),J)=1
1820 H=M( I),J)
1830 IF H=0 THEN 2180
1840 I=I+1
1850 X(I)=H
1860 IF X(I)=N THEN 2000
1870 K=1
1880 IF X(I)=X(K) THEN 1920
1890 K=K+1
1900 IF K=I THEN 1790
1901 GOTO 1880
1902 PRINT
1903 PRINT "CETTE BOUCLE CORRESPOND A LA SEQUENCE SUIVANTE"
1904 FOR J=K TO I-1
1905 G=0
1906 PRINT TAB(10);T$(X(J))
1907 FOR Z=1 TO C
1908 IF T(X(J),Z)<>E(X(J+1)) THEN 1914
1909 IF G=1 THEN 1913
1910 PRINT TAB(4);C$(Z)
1911 G=1
1912 GOTO 1914
1913 PRINT TAB(4);"OU";C$(Z)
1914 NEXT Z
```

```
1915 NEXT J
1916 PRINT TAB(10);T$(X(I))
1917 GOTO 1970
1920 PRINT
1922 PRINT
1925 PRINT TAB(4);"BOUCLE : ";
1930 FOR J=K TO I-1
1940 PRINT E(X(J));
1950 NEXT J
1960 PRINT E(X(I))
1965 GOTO 1902
1970 I=I-1
1980 IF I=0 THEN 2170
1990 GOTO 1790
2000 PRINT
2002 PRINT
2005 PRINT TAB(4);"CHEMIN : ";
2010 FOR J=1 TO I-1
2020 PRINT E(X(J));
2030 NEXT J
2040 PRINT E(X(I))
2045 GOTO 2062
2050 I=I-1
2060 IF I=0 THEN 2170
2061 GOTO 1790
2062 PRINT
2063 PRINT "CE CHEMIN CORRESPOND A LA SEQUENCE SUIVANTE"
2064 FOR J=1 TO I-1
2065 G=0
2066 PRINT TAB(10);T$(X(J))
2067 FOR K=1 TO C
2068 IF T(X(J),K)<>E(X(J+1)) THEN 2074
2069 IF G=1 THEN 2073
2070 PRINT TAB(4);C$(K)
2071 G=1
2072 GOTO 2074
2073 PRINT TAB(4);"OU";C$(K)
2074 NEXT K
2075 NEXT J
2076 PRINT TAB(10);T$(X(I))
2077 PRINT TAB(4);"FIN"
2078 GOTO 2050
2080 FOR K=1 TO J
2090 P(X(I),K)=0
2100 NEXT K
2110 I=I-1
2120 IF I=0 THEN 2170
2130 GOTO 1790
2140 IF J=C-1 THEN 2080
2150 J=J+1
2160 GOTO 1800
2170 END
2180 REM CAS OUBLIES
2190 FOR K1=2 TO C
2200 IF T(X(I),K1)=0 THEN 2225
2210 NEXT K1
2220 GOTO 2080
2225 IF T(X(I),1)<>0 THEN 2080
2226 T(X(I),K1)=1
2230 PRINT
2240 PRINT "LA SOUS- EQUENCE SUIVANTE EST A ANALYSER"
2250 PRINT "EST-CE UN CAS OUBLIE OU UN CAS QUI NE SE PRESENTE JAMAIS"
2255 IF I=1 THEN 2380
2260 FOR J=1 TO I-1
```

```
2270 G=Ø
2280 PRINT TAB(1Ø);T$(X(J))
2290 FOR K=1 TO C
2300 IF T(X(J),K)<>E(X(J+1)) THEN 236Ø
2310 IF G=1 THEN 235Ø
2320 PRINT TAB(4);C$(K)
2330 G=1
2340 GOTO 236Ø
2350 PRINT TAB(4);"OU";C$(K)
2360 NEXT K
2370 NEXT J
2380 PRINT TAB(1Ø);T$(X(I))
2390 PRINT TAB(4);C$(K1)
2400 IF K1=C THEN 246Ø
2410 K1=K1+1
2420 IF T(X(I),K1)=Ø THEN 244Ø
2430 GOTO 24ØØ
2440 T(X(I),K1)=1
2441 PRINT TAB(4);"OU";C$(K1)
2450 GOTO 24ØØ
2460 PRINT TAB(1Ø);"?"
2470 GOTO 2Ø8Ø
2500 DATA 7,4
2510 DATA Ø,2,Ø,2
2520 DATA Ø,1Ø,3,7
2530 DATA Ø,1Ø,3,7
2540 DATA Ø,1Ø,5,1Ø
2550 DATA Ø,4,Ø,4
2560 DATA Ø,5,2,2
2570 DATA 1,2,3,4,7,1Ø,5
2780 DATA "TØ","T1","T2","T1","T3","T2","T4"
2785 DATA "BI","A","B","C"
3000 :        PHASES  DECISIONS      CONDITIONS
3010 :                              ##  ##  ##  ##
3020 :          ##        ##        ##  ##  ##  ##
3030 :          ##        ##      FIN
*
```

ANALYSE DE LA TABLE SUIVANTE :

PHASES	DECISIONS	CONDITIONS			
		BI	A	B	C
1	TØ	Ø	2	Ø	2
2	T1	Ø	1Ø	3	7
3	T2	Ø	1Ø	3	7
4	T1	Ø	1Ø	5	1Ø
7	T	Ø	4	Ø	4
1Ø	T2	Ø	5	2	2
5	T4	FIN			

PAS DE BOUCLAGE INCONDITIONNEL

LES PHASES PRELEVEES SONT DANGEREUSES ET A ELIMINER

RECHERCHE DE TOUS LES CHEMINS ET BOUCLES ENTRE
LA PHASE INITIALE ET LES PHASES FINALES
AINSI QUE LES SOUS-SEQUENCES ABOUTISSANT AUX CAS OMIS

CHEMIN : 1 2 1Ø 5

CE CHEMIN CORRESPOND A LA SEQUENCE SUIVANTE

TØ
A
OUC
T1
A
T2
A
T4
FIN

BOUCLE : 2 1Ø 2

CETTE BOUCLE CORRESPOND A LA SEQUENCE SUIVANTE

T1
A
T2
B
OUC
T1

CHEMIN : 1 2 3 1Ø 5

CE CHEMIN CORRESPOND A LA SEQUENCE SUIVANTE

TØ
A
OUC
T1
B
T2
A
T2
A
T4
FIN

BOUCLE : 2 3 1Ø 2

CETTE BOUCLE CORRESPOND A LA SEQUENCE SUIVANTE

T1
B
T2
A
T2
B
OUC
T1

BOUCLE : 3 3

CETTE BOUCLE CORRESPOND A LA SEQUENCE SUIVANTE

T2
B
T2

CHEMIN : 1 2 3 7 4 1Ø 5

CE CHEMIN CORRESPOND A LA SEQUENCE SUIVANTE

TØ
A
OUC
T1
B
T2
C
T
A
OUC
T1
A
OUC
T2
A
T4
FIN

BOUCLE : 2 3 7 4 1Ø 2

CETTE BOUCLE CORRESPOND A LA SEQUENCE SUIVANTE

T1
B
T2
C
T3
A
OUC
T1
A
OUC
T2
B
OUC
T1

CHEMIN : 1 2 3 7 4 5

CE CHEMIN CORRESPOND A LA SEQUENCE SUIVANTE

```
        TØ
A
OUC
        T1
B
        T2
C
        T3
A
OUC
        T1
B
        T4
FIN
```

LA SOUS-SEQUENCE SUIVANTE EST A ANALYSER
EST-CE UN CAS OUBLIE OU UN CAS QUI NE SE PRESENTE JAMAIS

```
        TØ
A
OUC
        T1
B
        T2
C
        T3
B
        ?
```

CHEMIN : 1 2 7 4 1Ø 5

CE CHEMIN CORRESPOND A LA SEQUENCE SUIVANTE

```
        TØ
A
OUC
        T1
C
        T3
A
OUC
        T1
A
OUC
        T2
A
        T4
FIN
```

BOUCLE : 2 7 4 1Ø 2

```
CETTE BOUCLE CORRESPOND A LA SEQUENCE SUIVANTE
          T1
     C
          T3
     A
     OUC
          T1
     A
     OUC
          T2
     B
     OUC
          T1


     CHEMIN :  1    2  7  4  5

CE CHEMIN CORRESPOND A LA SEQUENCE SUIVANTE
          TØ
     A
     OUC
          T1
     C
          T3
     A
     OUC
          T1
     B
          T4
     FIN

LA SOUS-`EQUENCE SUIVANTE EST A ANALYSER
EST-CE UN CAS OUBLIE OU UN CAS QUI NE SE PRESENTE JAMAIS
          TØ
     B
          ?
 *
```

ANALYSE DE LA TABLE SUIVANTE :

PHASES	DECISIONS	CONDITIONS			
		BI	A	B	C
1	TØ	Ø	2	2	3
2	T1	4	Ø	Ø	Ø
3	T2	Ø	6	5	4
4	T1	7	Ø	Ø	Ø
5	T3	Ø	11	6	11
6	T4	Ø	9	11	8
7	T3	2	Ø	Ø	Ø
8	T5	Ø	7	9	5
9	T6	Ø	8	11	1Ø
1Ø	T1	7	Ø	Ø	Ø
11	T7	FIN			

RECHERCHE DES PHASES QUI DETERMINENT DES BOUCLES
INCONDITIONNELLLES ETVDES PHASES QUI Y CONDUISENT
INCONDITIONNELLEMENT

LA PHASE 2 APPARTIENT A UNE BOUCLE
INCONDITIONNELLE DE LONGUEUR 3
LA PHASE 4 APPARTIENT A UNE BOUCLE
INCONDITIONNELLE DE LONGUEUR 3
LA PHASE 1Ø MENE INC. A UNE BOUCLE
DE LONGUEUR 3
LA PHASE 7 APPARTIENT A UNE BOUCLE
INCONDITIONNELLE DE LONGUEUR 3

LES PHASES PRELEVEES SONT DANGEREUSES ET A ELIMINER

RECHERCHE DE TOUS LES CHEMINS ET BOUCLES ENTRE
LA PHASE INITIALE ET LES PHASES FINALES
AINSI QUE LES SOUS-SEQUENCES ABOUTISSANT AUX CAS OMIS

BOUCLE : 9 8 9

CETTE BOUCLE CORRESPOND A LA SEQUENCE SUIVANTE

T6

A

T5

B

T6

CHEMIN : 1 3 6 9 8 5 11

CE CHEMIN CORRESPOND A LA SEQUENCE SUIVANTE

TØ
C
T2
A
T4
A
T6
A
T5
C
T3
A
OUC
T7
FIN

BOUCLE : 6 9 8 5 6

CETTE BOUCLE CORRESPOND A LA SEQUENCE SUIVANTE

T4
A
T6
A
T5
C
T3
B
T4

LA SOUS-SEQUENCE SUIVANTE EST A ANALYSER
EST-CE UN CAS OUBLIE OU UN CAS QUI NE SE PRESENTE JAMAIS

TØ
C
T2
A
T4
A
T6
A
T5
A
?

CHEMIN : 1 3 6 9 11

CE CHEMIN CORRESPOND A LA SEQUENCE SUIVANTE

TØ
C
T2
A
T4
A
T6
B
T7
FIN

```
LA SOUS-SEQUENCE SUIVANTE EST A ANALYSER
EST-CE UN CAS OUBLIE OU UN CAS QUI NE SE PRESENTE JAMAIS
          TØ
    C
          T2
    A
          T4
    A
          T6
    C
          ?

    CHEMIN :  1    3  6  11

CE CHEMIN CORRESPOND A LA SEQUENCE SUIVANTE
          TØ
    C
          T2
    A
          T4
    B
          T7
    FIN

    BOUCLE :  8    9  8

CETTE BOUCLE CORRESPOND A LA SEQUENCE SUIVANTE
          T5
    B
          T6
    A
          T5

    CHEMIN :  1    3  6  8  9  11

CE CHEMIN CORRESPOND A LA SEQUENCE SUIVANTE
          TØ
    C
          T2
    A
          T4
    C
          T5
    B
          T6
    B
          T7
    FIN
```

CHEMIN : 1 3 6 8 5 11

CE CHEMIN CORRESPOND A LA SEQUENCE SUIVANTE

TØ
C
T2
A
T4
C
T5
C
T3
A
OUC
T7
FIN

BOUCLE : 6 8 5 6

CETTE BOUCLE CORRESPOND A LA SEQUENCE SUIVANTE

T4
C
T5
C
T3
B
T4

CHEMIN : 1 3 5 11

CE CHEMIN CORRESPOND A LA SEQUENCE SUIVANTE

TØ
C
T2
B
T3
A
OUC
T7
FIN

BOUCLE : 9 8 9

CETTE BOUCLE CORRESPOND A LA SEQUENCE SUIVANTE

T6
A
T5
B
T6

BOUCLE : 5 6 9 8 5

CETTE BOUCLE CORRESPOND A LA SEQUENCE SUIVANTE

T3
B
T4
A
T6
A
T5
C
T3

CHEMIN : 1 3 5 6 9 11

CE CHEMIN CORRESPOND A LA SEQUENCE SUIVANTE

TØ
C
T2
B
T3
B
T4
A
T6
B
T7
FIN

CHEMIN : 1 3 5 6 11

CE CHEMIN CORRESPOND A LA SEQUENCE SUIVANTE

TØ
C
T2
B
T3
B
T4
B
T7
FIN

BOUCLE : 8 9 8

CETTE BOUCLE CORRESPOND A LA SEQUENCE SUIVANTE

T5
B
T6
A
T5

CHEMIN : 1 3 5 6 8 9 11

CE CHEMIN CORRESPOND A LA SEQUENCE SUIVANTE

TØ
C
T2
B
T3
B
T4
C
T5
B
T6
B
T7
FIN

BOUCLE : 5 6 8 5

CETTE BOUCLE CORRESPOND A LA SEQUENCE SUIVANTE

T3

B

T4

C

T5

C

T3

LA SOUS-SEQUENCE SUIVANTE EST A ANALYSER
EST-CE UN CAS OUBLIE OU UN CAS QUI NE SE PRESENTE JAMAIS

TØ

C

T2

C

?

LA SOUS-SEQUENCE SUIVANTE EST A ANALYSER
EST-CE UN CAS OUBLIE OU UN CAS QUI NE SE PRESENTE JAMAIS

TØ

A

OUB

?

*

3. 7. Conclusion.

L'analyse à partir de la table a fourni :

- les boucles de la procédure. Les boucles inconditionnelles sont à supprimer par l'analyste ainsi que les phases qui y mènent inconditionnellement. D'autre part, l'analyste doit examiner les autres boucles afin de vérifier qu'au moins une des conditions dont dépend la sortie de la procédure de ces boucles sera vérifiée à un moment donné (au bout d'un temps fini) sinon la procédure bouclera sans fin;

- les sous-séquences de traitements et de conditions aboutissant aux cas omis. Le soin est laissé à l'analyste de préciser s'il s'agit de cas omis volontairement parce que ne survenant jamais ou de cas oubliés et par conséquent à introduire;

- les séquences de traitements et conditions possibles pour la procédure. Elles permettront à l'analyste de vérifier si la logique de la procédure est correcte, s'il n'y a pas d'incompatibilité (exemple, test d'une variable qui n'a jamais été positionnée) et si la procédure a été correctement mise en page.
  Elleslui permettent également de découvrir des problèmes tels une phase qui ne figure dans aucune séquence, ni boucle .... et d'autres.

En résumé, l'analyse automatique permet à l'analyste de vérifier la logique du problème défini (cas oubliés, cas d'incompatibilité, erreurs) et de vérifier que ce qui a été mis en page correspond bien au problème posé.

CHAPITRE IV.

LA TABLE SEQUENTIELLE : OUTIL D'AIDE A LA PROGRAMMATION.

Après mise en page, analyse et simplification d'une procédure grâce à une table séquentielle, cette procédure est prête à être implémentée. Entre la procédure sous forme de table et la procédure sous forme de programme, il reste un dernier aspect auquel doit pourvoir la table en tant qu'outil de synthèse : préparer et faciliter la tâche du programmeur chargé de l'implémentation.

4.1. Programmation manuelle.

4.1.1. Programmation directe à partir de la table.

Si la représentation d'une procédure sous forme de table séquentielle est familière au programmeur, il peut la programmer directement à partir de cette table.

Il lui suffit de programmer phase par phase ou encore ligne par ligne,en éclatant les différents traitements en les instructions correspondantes et en les articulant par les tests des conditions requises. Autrement dit, pour chaque phase, il doit d'abord traduire dans le langage de programmation choisi, le traitement associé à cette phase, ensuite programmer le test des différentes possibilités qui peuvent se présenter; en fonction du résultat de ces tests, il programmera des branchements vers d'autres parties du programme qui correspondent à l'implémentation d'autres phases. Ceci est possible si chaque phase programmée est référencée par une étiquette ou un numéro d'instruction (selon les possibilités offertes par le langage de programmation).

4.1.2. Programmation via un organigramme.

Dans certains cas, il est souhaité de convertir la table en organigramme : il se peut d'une part que l'utilisateur et/ou le programmeur le demande pour des raisons d'habitude et de convenance personnelle, mais d'autre part, comme nous pourrons le constater, le passage par un organigramme permet de programmer la procédure d'une manière avantageuse tant du point de vue clarté que du point de vue maintenance.

La conversion d'une table séquentielle en organigramme peut s'opérer de plusieurs manières; nous considérerons les 3 cas suivants :

1 - établir un organigramme qui ne soit qu'une retranscription de la table (§ 4.1.2.1)

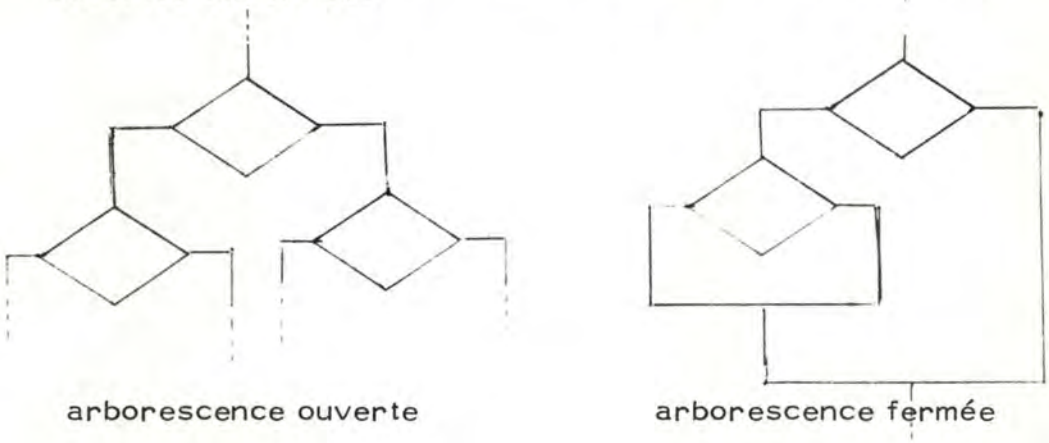
2 - établir un organigramme de structure classique (§ 4.1.2.2)

3 - établir un organigramme de structure classique et modulaire (§ 4.1.2.3).

Avant toute chose, précisons ce que nous entendons par organigramme de structure classique ou modulaire : tout organigramme est caractérisé par une structure

- chaînée
- répétitive
- alternative.

SI les alternatives ne donnent lieu qu'à des arborescences ouvertes ou fermées



arborescence ouverte

arborescence fermée

ALORS la structure de l'organigramme est dite classique .

SI l'organigramme est composé d'un ensemble de modules qui, lorsqu'ils sont parcourus, le sont toujours de leur racine à leur extrémité et qui sont, soit distincts, soit inclus les uns dans les autres

ALORS la structure de l'organigramme est modulaire.

4.1.2.1. Organigramme de retranscription de la table.

L'élaboration de l'organigramme de retranscription ne présente aucune difficulté. Il n'est que l'image de la table. Il s'obtient par retranscription de la table, ligne par ligne et point par point. Autrement dit, il suffit de reprendre les traitements qui sont éventuellement associés aux différentes lignes (phases) et de les articuler par les tests des conditions comme spécifié par la table.
Notons que lorsque nous programmons une procédure directement à partir d'une table sans passer par un organigramme, nous programmons implicitement à partir de l'organigramme de retranscription de cette table.

Exemple : Soit la table séquentielle :

PHASES	TRAIT.	B.I.	C_1	C_2	C_3	$\overline{C}_i$
1	-		2	2	3	4
2	T_1		5	-	-	6
3	-		-	4	-	5
4	T_1		2	5	-	6
5	T_2	6				
6	-		3	4	-	7
7	FIN					

Figure IV-1.

L'organigramme qui en est la retranscription est celui de la figure IV-2.

Comme nous pouvons le constater à partir de l'exemple, les organigrammes de retranscription ne sont généralement ni modulaires ni classiques.

S'ils sont faciles à établir, il n'en est pas moins vrai que leur structure devient rapidement un enchevêtrement inextricable de figurines et que leur clarté est mise en péril. Ils ne permettent ni une exploitation, ni une maintenance, ni une recherche d'erreurs aisées. De plus, il n'est pas possible de suivre le programme pas à pas lors de son déroulement; en effet, les phases successives qui composent la procédure n'y sont pas enregistrées et il est donc difficile de localiser à tout moment la phase en cours notamment lors de l'exécution d'un traitement commun.

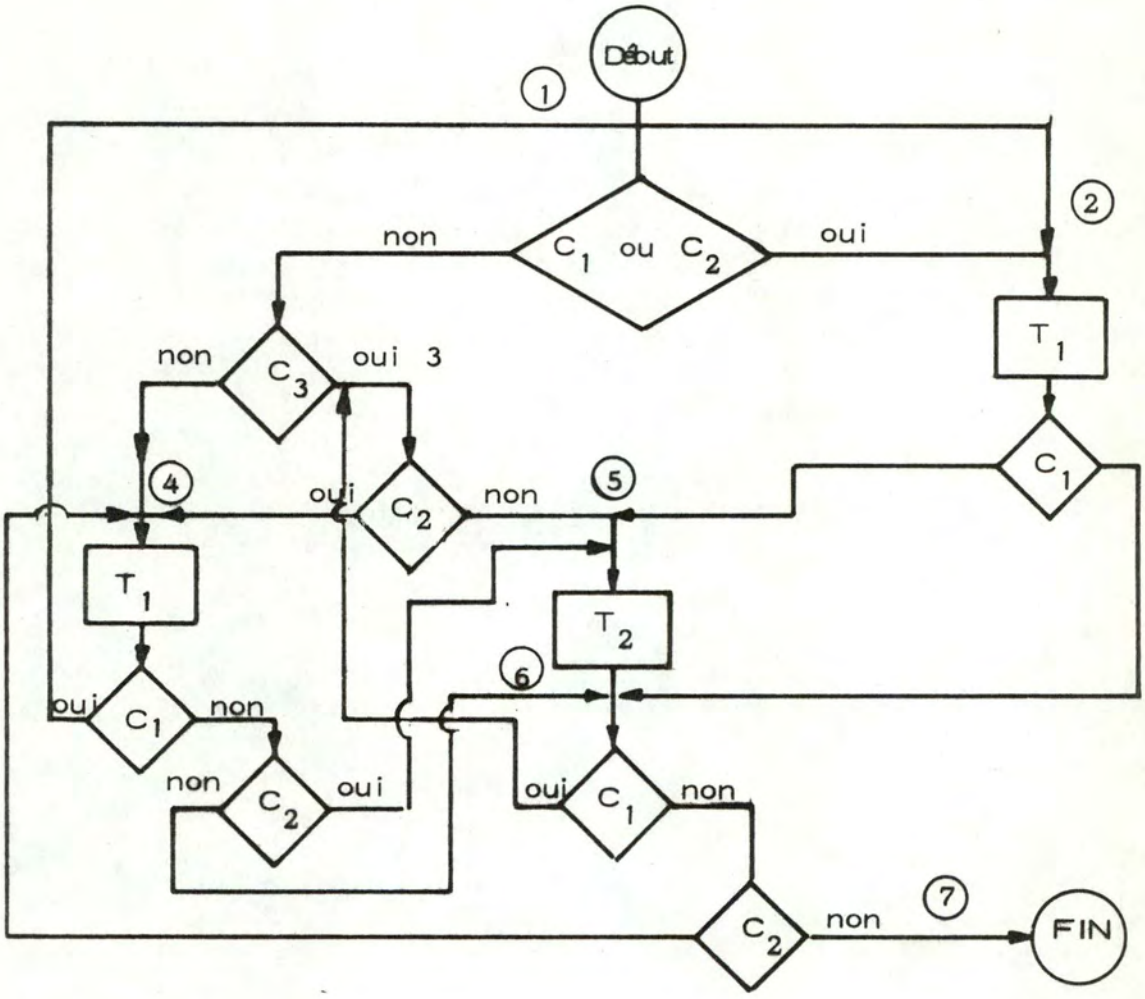
Si nous désirons éviter les inconvénients, nous adopterons de préférence l'organigramme modulaire ou classique.

4. 1. 2. 2. Organigramme classique.

Pour obtenir un organigramme de structure classique à partir d'une table séquentielle, l'idée de base est d'exécuter les opérations qui sont communes à plusieurs phases dans un seul et même module. Ceci est d'ailleurs conseillé au niveau de la programmation. Ces opérations peuvent être, soit les traitements, soit les tests de condition; nous ferons le raisonnement dans le cas des traitements.

De l'idée de n'associer qu'un seul module à chaque traitement, découle qu'il faudra, lors de l'exécution d'un traitement commun, connaître la phase qui est en cours.
Pour cela, deux possibilités :

1. faire réapparaître le paramètre de mémorisation d'état (E) que nous avions jugé non indispensable en logique



ORGANIGRAMME DE RETRANSCRIPTION

Figure IV-2

L'organigramme s'établit en associant à chaque traitement un module (celui-ci lors de l'implémentation correspondra par exemple à une sous-routine); ensuite

- si le traitement n'est jamais exécuté qu'au cours d'une seule phase, il suffit d'introduire à la suite du traitement, les tests des conditions requises, conformément à ce qui est spécifié dans la table; ce afin de déterminer la suite du déroulement
- si le traitement est commun à plusieurs phases, il faudra cette fois, à la suite du traitement, introduire le test de la variable E ou des aiguillages de discrimination de phases, suivi du test des conditions requises.

L'organigramme obtenu de cette manière est classique; il ne possède que des arborescences ouvertes.

Solution utilisant le paramètre d'état :

Puisque l'exécution de chacun des traitements s'opère dans un même module et qu'il faut distinguer la phase dans laquelle elle a lieu, une première solution est d'enregistrer l'état suivant en vue de savoir quel sera l'état présent lors du prochain traitement. Cet enregistrement se fera grâce à la variable d'état E.

Exemple : l'organigramme classique défini par la table de la figure V-1 et utilisant le paramètre E est celui de la figure V-3. Selon qu'on se trouve en l'état 1, 3 ou 6, après exécution du traitement vide, on s'orientera suivant les valeurs binaires de C_1, C_2, C_3 vers 4, 3, 2 à partir de 1; vers 2, 4, 3 à partir de 6; vers 4, 5 à partir de 4.

Solution utilisant les aiguillages de discrimination de phases :

Dans la solution précédente, le paramètre E discriminait tous les états ou phases. Il n'est toutefois pas indispensable que toutes les phases soient discriminées; il suffit que le soient celles qui possèdent un traitement commun.
De là cette deuxième solution qui consiste à discriminer par des aiguillages (binaires) seulement les phases nécessaires.

Le nombre d'aiguillages ainsi que leur organisation est fixé par le nombre le plus grand d'origines à discriminer pour chacun des traitements.

Exemple : reprenons, une fois encore, la table séquentielle de la figure V-1.

- les phases sans traitement sont au nombre de 3 (1, 3, 6)
- les phases avec T_1 sont au nombre de 2 (2, 4)
- les phases avec T_2 sont au nombre de 1 (5)

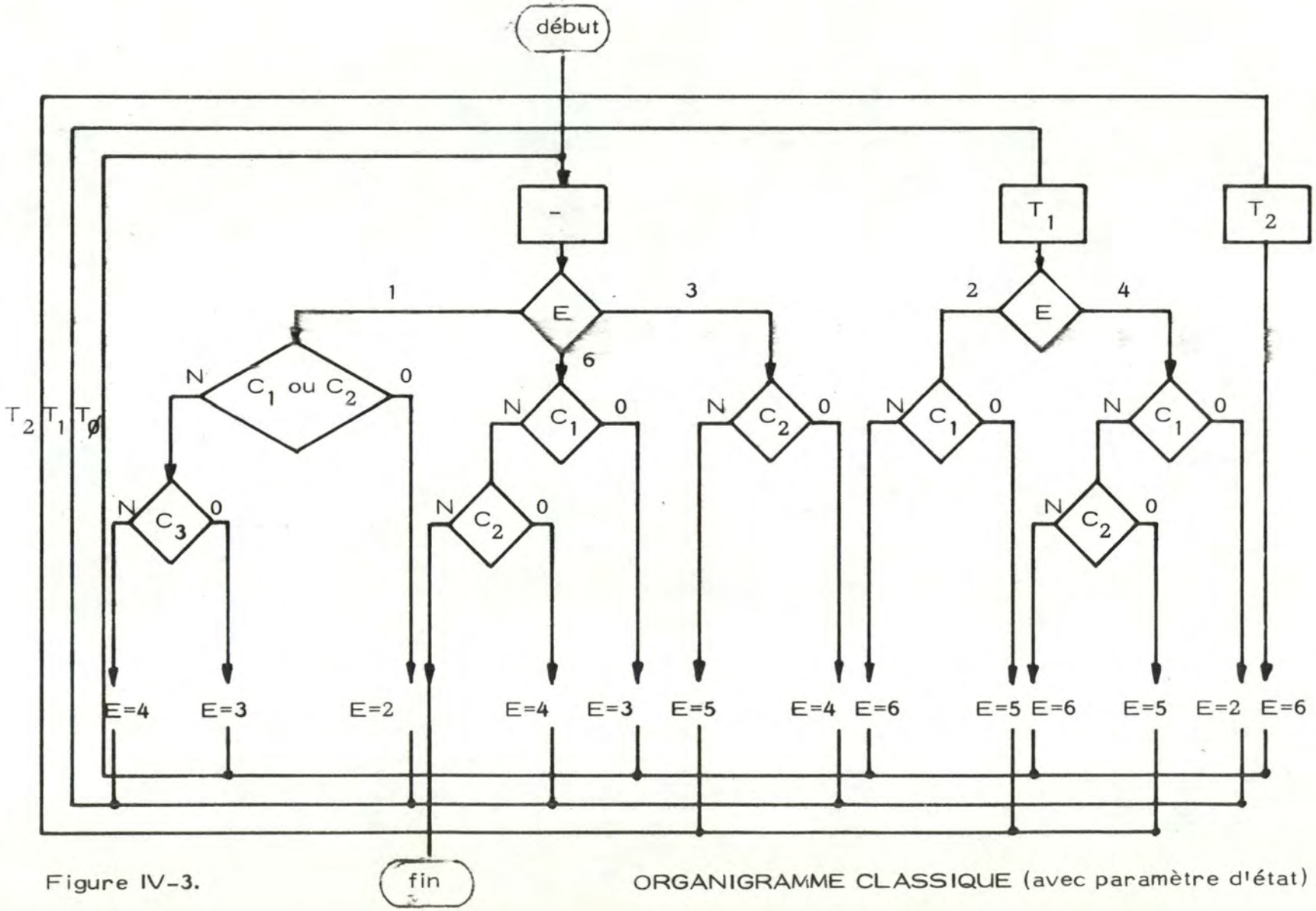


Figure IV-3. ORGANIGRAMME CLASSIQUE (avec paramètre d'état)

Deux aiguillages S_1 et S_2 suffiront donc :

S_1	S_2	PHASES	TRAIT.	BI.	C_1	C_2	C_3	$\overline{C}_i$
0	0	1	-		2	2	3	4
0	-	2	T_1		5	-	-	6
1	0	3	-		-	4	-	5
1	-	4	T_1		2	5	-	6
-	-	5	T_2	6				
0	1	6	-		3	4	-	7
-	-	7	FIN					

L'organigramme obtenu est celui de la figure V-4.

En résumé, dans les 2 organigrammes nous voyons que chaque traitement est réalisé pour les états qui lui sont associés et que lorsque le traitement est terminé, un test soit de la variable E, soit des aiguillages, est opéré pour déterminer l'état présent. Le test sera suivi d'un saut vers le trait caractérisant l'état suivant, en fonction des résultats des tests des conditions requises. Le cycle se reamorce alors avec ce nouveau traitement.

L'organigramme de structure classique, grâce à la décomposition en blocs sur base des traitements, assure au programmeur une structure favorable à la maintenance, l'exploitation, la recherche et la localisation des erreurs. Ainsi supposons, par exemple, que dans l'organigramme V-3 ou V-4, des traitements complémentaires à T_1 et différents selon la phase, T'_1, T''_1, T'''_1, doivent être envisagés, il n'y aura aucune difficulté à les y intégrer. De plus, cette structure permet de suivre le déroulement de la procédure implémentée, pas à pas.

4.1.2.3. Organigramme modulaire et classique.

Cet organigramme modulaire et classique non seulement va rendre compte de l'ordre qui affecte la procédure, des différents traitements qui la composent et de la manière dont ils sont subordonnés aux conditions, mais permet également de connaître à tout instant l'état ou phase en cours grâce à une variable d'état E. Celle-ci varie lors du déroulement de la procédure, comme les états qui se succèdent.

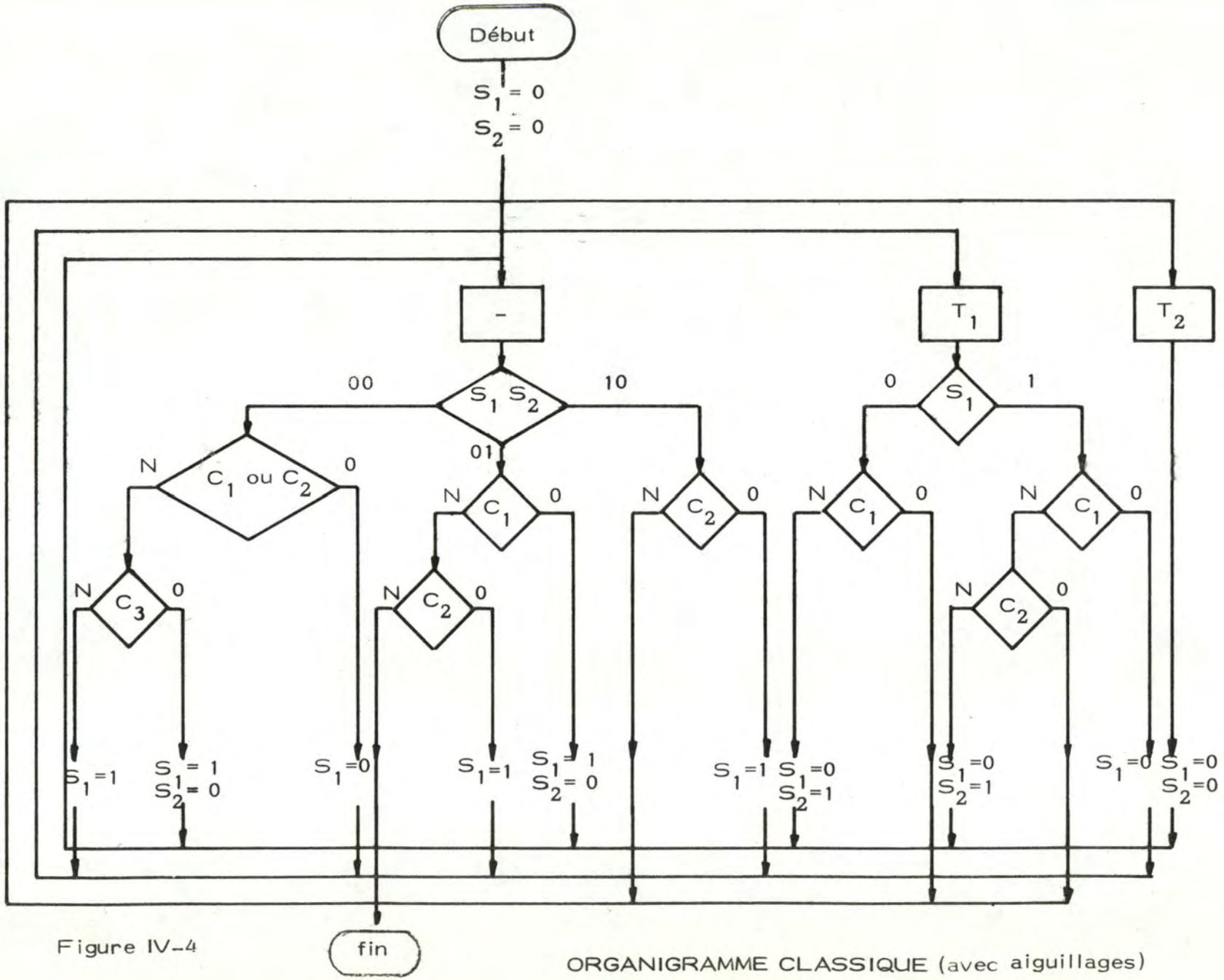


Figure IV-4

ORGANIGRAMME CLASSIQUE (avec aiguillages)

Nous voyons donc également réapparaître le paramètre de mémorisation d'état (E). C'est grâce à lui que nous pourrons à partir d'une table, établir un organigramme non seulement classique mais aussi modulaire.

Cet organigramme est constitué de deux parties :

1. Le système de sélection de l'état présent (test de la valeur du paramètre d'état E).
2. L'ensemble des états ou phases examinés en séquence et durant lesquels, dans l'ordre
   a. un traitement est éventuellement exécuté
   b. des tests de conditions sont opérés
   c. la mise à jour de l'état présent est effectuée.

Les modules qui composent cet organigramme correspondent à la traduction des différentes phases qui composent la procédure et sa structure classique se justifie par le fait qu'il ne renferme que des arborescences fermées. Remarquons, en outre, son caractère itératif : à chaque changement d'état, il est reparcouru.

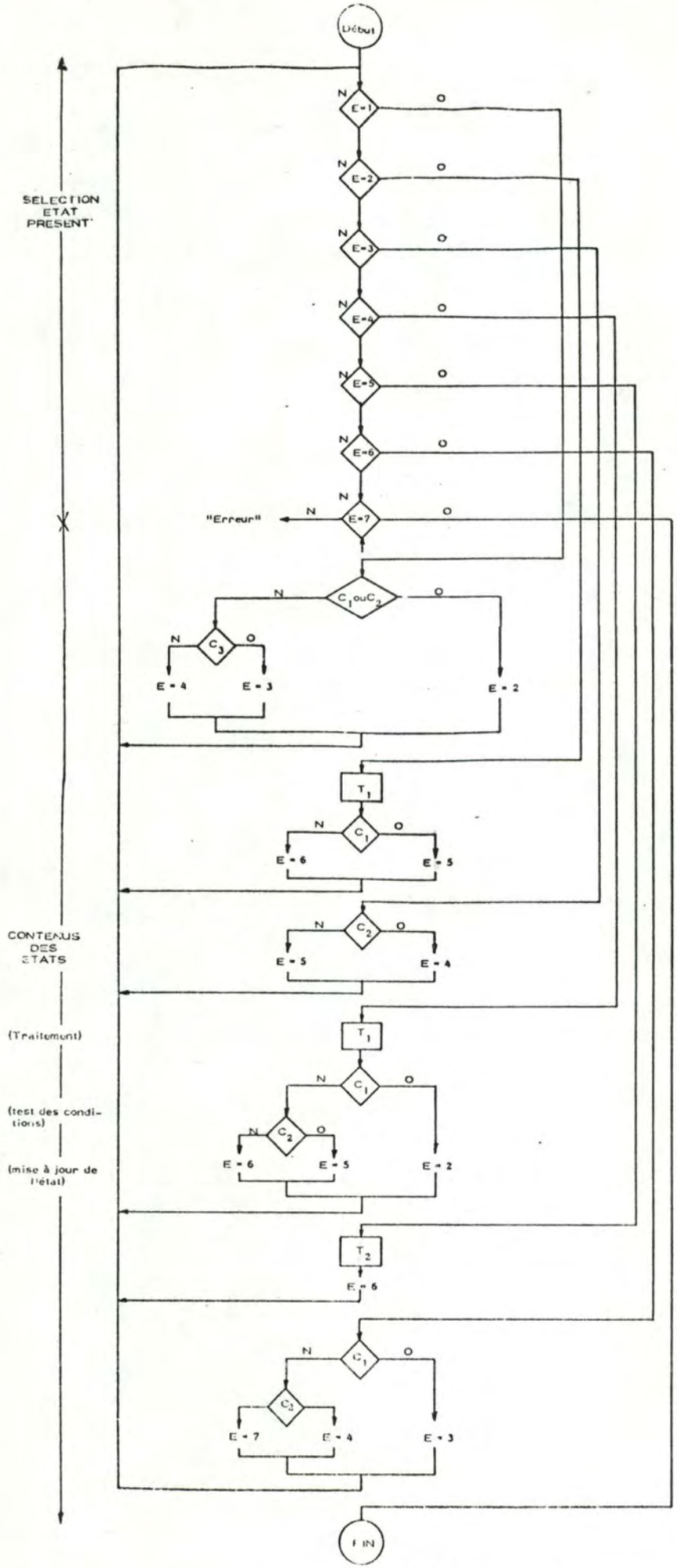
Exemple : L'organigramme modulaire et classique engendré par la table V-1 est celui de la figure V-5.

En dehors de la clarté et de la généralité de conception du programme qui en découle, cette solution, tout comme celle du paragraphe précédent, assure une grande facilité de maintenance. L'avantage le plus évident est celui qui permet de connaître à tout instant, grâce à la variable E, la situation logique du programme; ceci est très utile en cas d'anomalie ou lorsqu'il s'agit de situer des points de reprise.

4.1.2.4. Conclusion.

Les trois organigrammes que nous avons considérés, s'établissent de manière bien précise, chacun. Inutile de souligner que seuls l'organigramme classique et l'organigramme modulaire présentent des avantages certains. Ils permettent d'éviter deux choses :

1. lorsqu'un analyste établit un organigramme, il est guidé par quelques principes de logique et d'esthétique; il possède une méthode souvent très personnelle et a acquis un style parfois très original
2. les organigrammes rendent compte des enchaînements logiques d'un problème mais plus le problème est complexe, plus les enchaînements sont nombreux et l'ensemble acquiert rapidement une clarté douteuse



ORGANIGRAMME MODULAIRE ET CLASSIQUE

Figure IV-3

par le fait qu'ils ne sont pas considérés comme outil de synthèse mais uniquement comme document de base pour le programmeur; ils sont établis non plus lors de la mise en page d'une procédure mais après sa mise en page, son analyse et sa simplification.

Grâce à la table séquentielle, nous avons donc éclaté le problème de la synthèse, de là l'obtention d'un meilleur résultat au niveau de l'organigramme, d'un meilleur document de travail pour le programmeur.

4.2. Programmation semi-automatique.

La programmation semi-automatique s'applique à des cas très particuliers. Nous l'avons dénommée ainsi parce qu'elle consiste non pas à programmer la procédure même qui est enregistrée dans la table, ce qui implique la programmation d'une multitude de tests de conditions suivis de branchements, mais à implémenter la table séquentielle elle-même ainsi qu'un programme de consultation de cette table.

Nous avons vu au chapitre II paragraphe 2.4.1 que dans certains cas la procédure était définie par un ensemble de séquences que nous avons appelées "séquences de base". Ces procédures sont toujours articulées par des conditions de nature extrinsèque.

Dans les cas où les procédures sont enregistrées dans une table condensée, leur implémentation est aisée. En effet, il suffit d'établir un programme de consultation de la table qui, au moment de l'exécution, se trouvera soit en mémoire centrale, soit sur un fichier à accès direct (disques). Les données de ce programme de consultation ne correspondent à rien d'autre qu'à la détermination de celles parmi les conditions de la procédure qui sont vérifiées et non vérifiées. Le programme consultera la table en sautant d'une ligne à l'autre, c'est-à-dire d'une phase à l'autre comme suit :

- lorsqu'il saute à une nouvelle ligne, il détermine à partir des données, celle des colonnes qui correspond à la condition qui est vérifiée ou qui correspond au cas où aucune condition ne l'est; l'élément de la table définie par la ligne et cette colonne indique à quelle nouvelle ligne le programme doit sauter
- initialement, le programme se branche à la ligne correspondant à la phase initiale de la procédure
- le programme s'arrête lorsqu'il a atteint une phase finale après avoir exécuté le traitement associé à cette phase; traitement que le programmeur aura eu soin de programmer également.

Ce type de programmation sera utilisé pour l'implémentation de la procédure définie par l'application IV du chapitre V. Il possède possède cependant un inconvénient majeur, la place occupée par la table.

4.3. Programmation automatique.

Nous attirons l'attention du lecteur sur le fait que nous n'avons examiné, ni à fond, ni complètement le problème de la programmation automatique à partir d'une table séquentielle. Nous nous bornerons à suggérer quelques idées pour lancer le problème.

La programmation automatique de tables séquentielles nécessite avant toute chose, le choix d'une syntaxe permettant d'exprimer ces tables d'une manière standard.

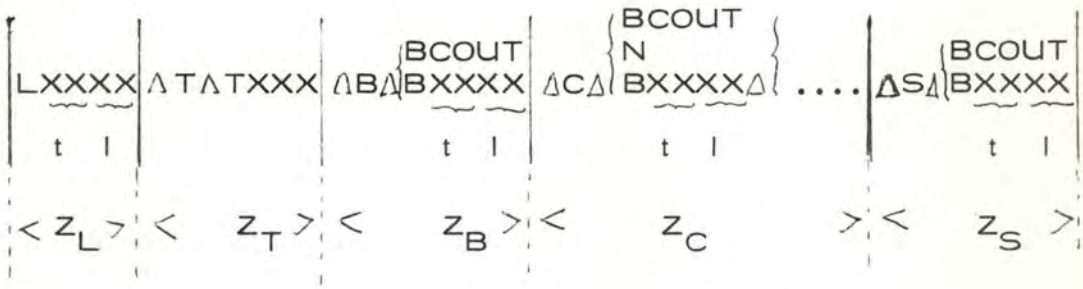
Il s'agit de trouver une forme d'expression commune aux tables condensées et détaillées.

Le format proposé pour chaque ligne de la table est celui de la figure IV-10. Les zones qui le composent sont les suivantes :

- Z_L : identifie une ligne d'une table et est caractérisée par le symbole L en en-tête. La sous-zone t identifie la table; la sous-zone l identifie la ligne (généralement il s'agit de la référence numérotée de la ligne dans la table). Ces 2 sous-zones occupent 2 positions chacune.

- Z_T : cette zone est facultative; elle n'apparaît que si la ligne spécifie un traitement à exécuter. Elle est caractérisée par le symbole T en en-tête. Le traitement à exécuter s'identifie au moyen d'une référence TXXXX (où X représente un chiffre de ø à 9).

- Z_B : le symbole B en en-tête signale la présence de cette zone qui est également facultative; elle n'apparaît que si la ligne dont il est question



(Δ = vide ou caractère "blanc")

figure IV - 10

donne la main inconditionnellement à une autre; celle-ci s'identifie par BXXXX (t l) ou par BCOUT si elle est associée à une phase finale.

- Z_C : cette zone est caractérisée par le symbole C en en-tête; celui-ci précède une série d'informations dont le nombre dépend du nombre de conditions binaires successives à tester.

Ces informations sont :

- soit des N indiquant que le test ne doit pas être effectué
- soit l'expression BCOUT (fin de procédure ou de programme) ou B$\underbrace{XX}_{t}\underbrace{XXX}_{l}$ indiquant à quelle table et à quelle ligne il faut brancher si le test de la condition correspondante est positif et si tous les tests des conditions précédentes ont été négatifs.

- Z_S : zone dont la présence est caractérisée par le symbole S en en-tête; le symbole est suivi de l'expression BCOUT ou B$\underbrace{XX}_{t}\underbrace{XXX}_{l}$ indiquant l'adresse d'un branchement (fin de programme ou nouvelle ligne) si toutes les conditions sont non testables (N dans Z_C) ou testées négativement.

Remarque : les différentes zones doivent toujours apparaître dans l'ordre indiqué.

Exemples : considérons les tables détaillées et condensées des figures IV-11 et IV-12. Leur traduction sous forme standard est donnée par les figures IV-13 et IV-14 respectivement.

PHASES	TRAIT.	B.I.	C_1		C_2	
			0	1	0	1
1	TØØ1		2	3		
2	TØØ2	4				
3	TØØ3	4				
4			5	6		
5	TØØ5	OUT				
6					7	1
7		OUT				

figure IV-11

PHASES	TRAIT	B.I.	C_1	C_2	C_3	C_4	C_5	C_6	$\overline{C_i}$
1	TØØ1		2						1
2							5	4	3
3	TØØ2	4							
4	TØØ1				7				6
5	TØØ3	25							
6	TØØ4	22							
7		21							

figure IV-12

Remarques : - dans la table II-12, les numéros supérieurs à 7 référencent les phases d'une deuxième table

- les numéros des lignes se confondent avec les références numérotées des phases.

```
BØ1Ø1 T TØØ1 C BØ1Ø3 N S BØ1Ø2
BØ1Ø2 T TØØ2 B BØ1Ø4 C N N
BØ1Ø3 T TØØ3 B BØ1Ø4 C N N
BØ1Ø4 C BØ1Ø6 N S  BØ1Ø5
BØ1Ø5 T TØØ5 B BCOUT C N N
BØ1Ø6 C N BØ1Ø1 S BØ1Ø7
BØ1Ø7 B BCOUT C N N
```

figure IV-13

```
BØ1Ø1 T T TØØ1 C BØ1Ø2 N N N N N S BØ1Ø1
BØ1Ø2 C N N N N N BØ1Ø5 BØ1Ø4 S BØ1Ø3
BØ1Ø3 T T TØØ2 B BØ1Ø4 C N N N N N N
BØ1Ø4 T T TØØ1 C N N BØ1Ø7 N N N S BØ1Ø6
BØ1Ø5 T T TØØ3 B BØ2Ø5 C N N N N N N
BØ1Ø6 T T TØ4 B BØ2Ø2 C N N N N N N
BØ1Ø7 B BØ2Ø1 C N N N N N N
```

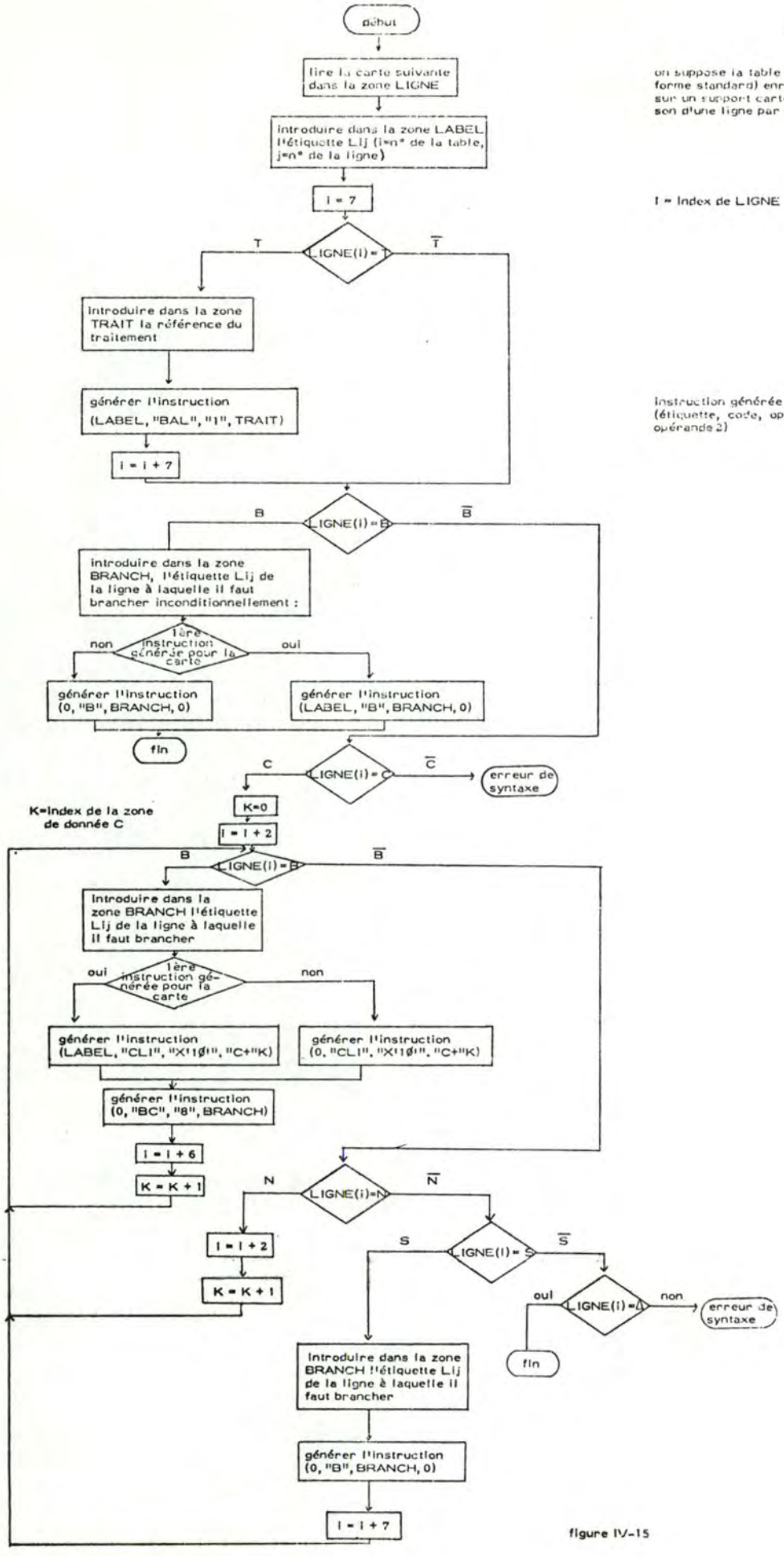
figure IV-14

Soit une procédure enregistrée dans une table séquentielle et soit cette table exprimée ligne par ligne selon le format décrit ci-dessus. Cette procédure va être confiée, sous cette forme, à un programme générateur qui va l'implémenter sous forme d'un programme ASSEMBLEUR.

Examinons quelque peu ce programme générateur et la façon dont il procède pour engendrer les programmes source assembleur. Son squelette est donné par la figure IV-15. Il faut noter que :

- la zone C est la zone des données "Condition" dans laquelle chaque condition correspond à un octet. Cette zone est remplie soit initialement, soit en cours d'exécution. Chaque octet contient soit la valeur X'1Ø' soit X'ØØ' selon que la condition associée est ou non vérifiée

- les différents traitements qui composent la procédure sont à programmer séparément; chacun d'eux sera étiquetté par la référence qui l'identifie dans la table.



on suppose la table (sous forme standard) enregistrée sur un support carte à raison d'une ligne par carte

I = index de LIGNE

instruction générée : (étiquette, code, opérande 1, opérande 2)

figure IV-15

Le programme source généré pour la procédure enregistrée dans la table IV-12 est :

```
L11     BAL  1,TØØ1
        CLI  X'1Ø',C
        BC   8,L12
        B    L11
L12     CLI  X'1Ø',C+4
        BC   8,L15
        CLI  X'1Ø',C+5
        BC   8,L14
        B    L13
L13     BAL  1,TØØ2
        B    L14
L14     BAL  1,TØØ1
        CLI  X'1Ø',C+2
        BC   8,L17
        B    L16
L15     BAL  1,TØØ1
        B    L25
L16     BAL  1,TØØ3
        B    L22
L17     B    L21

-----------------------

C       DC   7X'ØØ'
```

CHAPITRE V - APPLICATIONS

Le but de ce chapitre est d'une part, présenter quelques procédures qui se prêtent particulièrement bien à la mise en page sous forme de tables séquentielles, d'autre part, familiariser quelque peu le lecteur à ce nouvel outil qu'est la table de décision séquentielle.

Nous nous limiterons toutefois au premier aspect de la synthèse : la mise en page; en effet, elle seule présente un caractère inconnu. L'analyse et la simplification quant à elles, n'offrent aucune difficulté. Elles ont été détaillés au cours des chapitres précédents; l'analyse peut être réalisée automatiquement grâce au programme que nous avons développé.
Pour l'application IV (Evaluation du langage oral d'handicapés mentaux), nous irons jusqu'à l'implémentation proprement dite car elle a été réalisée concrètement.

Remarquons que les tables qui figureront dans ce chapitre, sont toutes de forme condensée (ou mixte) : forme la plus avantageuse de la table.

Les applications qui seront développées dans ce chapitre sont les suivantes :

- APPLICATION I : Analyse des langages réguliers par table séquentielle.
- APPLICATION II : Scanner sous forme de table séquentielle.
- APPLICATION III : Analyse syntaxique et sémantique des programmes BASIC par tables séquentielles.
- APPLICATION IV : Evaluation du langage oral d'handicapés mentaux.
- APPLICATION V : Procédure de gestion des échanges en télétraitement par caractères de contrôle.

× × ×

APPLICATION I - Analyse des langages réguliers par table séquentielle.

1. Rappel.

A. Définition d'une grammaire.

Reprenons la terminologie de Chomsky : une grammaire est par définition un quadruplet

$$G = \{ V_T , V_{NT} , S , R \}$$

avec
- V_T ensemble fini d'éléments appelés symboles terminaux
- V_{NT} ensemble fini d'éléments appelés symboles non terminaux
- $S \in V_{NT}$, symbole distingué
- R ensemble fini de règles de la forme

$$\alpha \longrightarrow \beta$$

avec $\alpha \in V^+ = (V_T \cup V_{NT})^+$

$= \{$chaînes obtenues par concaténation d'éléments de V$\}$

$\beta \in V^* = V^+ \cup \{\varepsilon\}$

chaîne vide

et α contenant au moins un symbole non terminal.

Exemple : soit la grammaire G_1 définie par :

$G_1 \equiv V_T = \{a, b, c\}$

$V_{NT} = \{S\}$

S symbole distingué

règles
- $S \longrightarrow a\,S\,a$
- $S \longrightarrow b\,S\,b$
- $S \longrightarrow c$

B. Interprétation d'une grammaire.

Une grammaire G s'interprête de la manière suivante :

1. Soit $\alpha, \beta \in V^*$: β est directement déductible de α

$\alpha \Longrightarrow \beta$

si et seulement si

$\exists$ une règle $\varphi \longrightarrow \psi \mid \alpha \equiv \alpha_1 \varphi \alpha_2$

et $\beta \equiv \alpha_1 \psi \alpha_2$ avec $\alpha_1, \alpha_2 \in V^*$

2. Soit $\alpha, \beta \in V^*$: β est déductible de α

$\alpha \Longrightarrow^* \beta$

si et seulement si

$\alpha = \beta$ ou $\exists\, \alpha_i \mid \alpha_i \Longrightarrow \beta$ et $\alpha \Longrightarrow^* \alpha_i$

3. Le langage engendré par une grammaire G est par définition

$L(G) = \{\alpha \mid \alpha \in V_T^*$ et $S \Longrightarrow^* \alpha\}$

Exemple : soit la grammaire G_1 définie plus haut nous avons

$bSa \Longrightarrow bca$

$SabSb \Longrightarrow^* cabaSab$

en effet $SabSb \Longrightarrow cabSb \Longrightarrow cabaSab$

$L(G_1) = \{\alpha\, c\, \bar{\alpha} \mid \alpha \in \{a, b\}^*$

et $\bar{\alpha}$ symétrique de $\alpha\}$

C. Grammaire régulière.

Chomsky définit 4 types de grammaire, parmi lesquelles les grammaires régulièresdont les règles sont de la forme

$$A \longrightarrow Ba$$
$$\text{ou } A \longrightarrow a \quad \text{avec} \quad A, B \in V_{NT}$$
$$a \in V_T$$

2. Analyse des langages réguliers par table séquentielle.

A. Problème général de l'analyse.

Il s'agit de déterminer, étant donné une grammaire, si une chaîne de symboles terminaux appartient ou non au langage engendré par cette grammaire.

Les algorithmes d'analyse peuvent procéder de deux manières :

Top-down : Partir du symbole distingué et se servir des règles de la grammaire de manière à déduire la chaîne des symboles terminaux à analyser.

Bottom-up : Partir de la chaîne des symboles terminaux à analyser et se servir des règles de la grammaire de manière à réduire (opération inverse de la déduction) la chaîne au symbole distingué.

Exemple : soit la grammaire G_2 (number) définie par les règles :

< number > :: = < no >

< no > :: = < no > < digit >

< no > :: = < digit >

< digit > :: = 0 / 1 / 2 / 3 / 4 / 5 / 6 / 7 / 8 / 9

(nous avons utilisé le formalisme de Backus-Naur pour définir cette grammaire).

35 ∈ L (G2) ?

analyse top-down : < number > ⇒ < no > ⇒ < no >

< digit > ⇒ < no > 5 ⇒ < digit > 5

⇒ 35

analyse bottom-up : 35 ⇐ < digit > 5 ⇐ < digit > < digit >

⇐ < no > < digit > ⇐ < no >

⇐ < number >

Remarque.

Lors de chaque réduction directe dans l'analyse bottom-up, on a cherché, en partant de la gauche, s'il existait une sous-chaîne se terminant au point où on en était arrivé et qui était partie droite d'une règle de la grammaire; si oui, on réduisait directement cette sous-chaîne à la partie gauche de la règle. Cette sous-chaîne est appelée poignée (soulignée dans l'exemple 35 ⟹ < digit > 5).

B. Table séquentielle définie par une grammaire régulière.

Résolvons le problème de l'analyse des grammaires et langages réguliers.

Nous allons construire, étant donné une grammaire régulière, une table séquentielle qui pour une chaîne quelconque de symboles terminaux, détermine si elle appartient ou non au langage engendré par cette grammaire.

Soit S la phase initiale de la table.

Les éléments de la table définis par la grammaire sont :

1. Tout symbole non terminal de la grammaire définit une phase;
2. Tout symbole terminal t de la grammaire définit une condition ou en-tête de la table : "symbole lu = t";
3. a. Toute règle du type A ⟶ Ba définit une transition phase présente, phase suivante pour la condition d'entrée "symbole lu = a":

$$T(B, a) = A$$

phase présente phase suivante

b. Toute règle du type A ⟶ a définit une transition phase présente, phase suivante pour la condition d'entrée "symbole lu = a":

$$T(S, a) = A$$

phase initiale phase suivante

A ces éléments viennent s'ajouter :

1. La condition en en-tête : "symbole lu = * " ou * représente le symbole qui caractérise la fin de chaîne; il s'agit en général du symbole " ♭ " à condition que celui ne soit pas déjà symbole terminal.

2. a. Deux phases de fin de séquence :

F (fin) : phase qui conclut de l'appartenance de la chaîne terminale d'entrée;

E (erreur) : phase qui conclut de la non appartenance.

b. A ces 2 phases sont associées les transitions :

$T(E_i, *) = E \qquad \forall E_i$ phase $\neq E, F, Z$

$T(Z, *) = F$

3. Le traitement à réaliser au cours de chacune de ces phases (autre que E et F) est "lire le symbole suivant dans la chaîne d'entrée ".

Exemple : soit la grammaire G(2) définie par :

$V_{NT} = \{U, V, Z\}$

$V_T = \{0, 1\}$

Z symbole distingué

règles

$Z \longrightarrow U0$

$Z \longrightarrow V1$

$U \longrightarrow Z1$

$U \longrightarrow 1$

$V \longrightarrow Z0$

$V \longrightarrow 0$

Il lui correspond la table séquentielle :

PHASES	DECISIONS	SYMBOLE LU 0	1	*
S	Lire symbole	V	U	E
U	Lire symbole	Z		E
V	Lire symbole		Z	E
Z	Lire symbole	V	U	F
E	N'appartient pas; Fin			
F	Appartient ; Fin			

La table ainsi obtenue n'est pas complète; en effet, il manque les transitions :

$$T(U, 1) = ?$$
$$T(V, 0) = ?$$

Ces cases vides sont à examiner : avons-nous omis de traduire une règle de la grammaire ?

Si oui : complèter avec la phase suivante requise;

Si non : la phase suivante est E.

Remarque :

Il est important de noter que cette analyse par table telle que nous l'avons définie n'est possible que pour autant que toutes les parties droites des règles soient uniques; si ce n'est pas le cas, il existe un artifice qu'il serait trop long d'exposer ici et grâce auquel il y a moyen malgré tout de construire une table d'analyse.

C. Algorithme d'analyse utilisé par la table.

L'algorithme d'analyse que représente la table procède "bottom-up"; en effet, en chaque phase de la séquence, sauf en phase initiale, la poignée est le nom de la phase présente concaténé au symbole lu, sauf si ce dernier est le symbole de fin de chaîne.

Cet poignée est réduite directement au symbole terminal qui est la phase suivante définie par la table pour la phase présente et le symbole lu, à moins que cette phase suivante ne soit E ou F auquel cas l'analyse est terminée.

Soit la table ci-dessus et la chaîne d'entrée :

101001*

Phases présentes	Chaîne qui reste à lire	Poignée	Phases suivantes
S	101001	1	U
U	01001	U0	Z
Z	1001	Z1	U
U	001	U0	Z
Z	01	Z0	V
V	1	V1	Z
Z	*	-	F
F	↑ symbole lu		

L'analyse de cette chaîne terminale a permis d'obtenir la déduction :

$$Z \Longrightarrow V1 \Longrightarrow Z01 \Longrightarrow U001 \Longrightarrow Z001 \Longrightarrow U01001 \Longrightarrow 101001$$

ou encore $Z \Longrightarrow^{*} 101001$ et de conclure de l'appartenance de la chaîne au langage engendré par la grammaire.

Cette analyse est simple en raison de la nature simple des règles des grammaires régulières :

- un non-terminal ne peut apparaître que comme 1er symbole d'une partie droite, par conséquent, la phase initiale essaye de réduire directement le 1er symbole de la chaîne à un non-terminal;
- ce non-terminal est la phase suivant la phase initiale pour le 1er symbole de la chaîne (si elle n'est ni E, ni F) et ceci revient à appliquer la règle de grammaire

  Phase suivante ⟶ 1er symbole
  (non terminal) (terminal)

- les phases suivantes de la séquence essayent de réduire directement le symbole non terminal déterminé par la phase présente et le symbole lu, à un non terminal. Ce non-terminal est la phase suivante définie par la table pour cette phase présente et ce symbole lu (si elle est différente de E et F).
  Ceci revient à appliquer la règle :

  Phase suivante ⟶ Phase présente symbole lu
  (non terminal) (non terminal) (terminal)

D. Table séquentielle et automate.

Cet algorithme d'analyse développé dans la table est identique à celui formalisé dans la théorie des langages par la notion d'automate.

Pourquoi résoudre le problème de l'analyse par table séquentielle plutôt que par un automate ?

Le procédé d'analyse est le même, seules les présentations sont différentes. La présentation sous forme de table est plus concise et moins confuse lorsque les ensembles V_T et V_{NT} sont conséquents. Elle permet de fusionner les phases équivalentes (voir théorie générale) plus facilement qu'il n'est possible de fusionner les états équivalents de l'automate. De plus, la table permet de découvrir les cas oubliés ou erreur alors que l'automate ne le permet pas.

APPLICATION II - Scanner sous forme de table séquentielle.

1. Le Scanner.

Le scanner est la partie la plus simple d'un compilateur, appelée parfois analyseur lexical. Il scanne les caractères du programme source de gauche à droite et construit les symboles du programme (identificateurs, entiers, mots-clé). Il abandonne les commentaires.

Les symboles construits sont transmis à une autre partie du compilateur : l'analyseur syntaxique. Le scanner est généralement une sous-routine (SCAN) de ce dernier et appelé chaque fois que l'analyseur syntaxique requiert un nouveau symbole.

Les symboles sont transmis à l'analyseur syntaxique sous une forme interne de longueur fixe : par exemple, la représentation interne d'un identificateur est un nombre entier fixé, la représentation interne d'un entier, un autre entier fixé; tous les identificateurs auront même nombre de représentation interne. Ce sont ces nombres qui sont transmis à l'analyseur syntaxique. Le symbole lui-même est cependant utilisé dans d'autres parties du compilateur, c'est pourquoi il doit être stocké quelque part. La sous-routine SCAN aura donc 2 sorties :

- l'une est la représentation interne du symbole;
- l'autre le symbole lui-même ou un pointeur vers lui.

Exemple :

Symbole	Représentation interne
indéfini	0
identificateur	1
entier	2
BEGIN (mot-clé)	3
/	4
+	5
-	6

*	7
(	8
)	9
END	10
=	11

Soit le morceau de programme :

BEGIN A = BC + D END

les sorties de SCAN seront successivement

3 , 'BEGIN'
1 , 'A'
11 , '='
1 , 'BC'
5 , '+'
1 , 'D'
10 , 'END'

2. Reconnaissance des symboles par table séquentielle.

Afin de donner une allure concrète à l'application, considérons le langage de programmation Algol 60. (*)

Les symboles de base du langage sont :

- les identificateurs;
- les mots-clé : BEGIN, IF, COMMENT .... (qui forment un sous-ensemble de l'ensemble des idenficateurs);
- les délimiteurs simples : (,) , + , $\leq$, = ...
- le délimiteur double : :=
- les entiers
- les strings.

Remarque :

Souvent la distinction entre les symboles de base et les constructions de niveau plus élevé est vague. Par exemple, nous pouvons considérer, et les nombres entiers, et les nombres réels, comme des symboles de base ou seulement les nombres entiers et les nombres réels comme des constructions de niveau plus élevé.

(*) Revised Report on the Algorithmic language Algol 60-CACM-6, 1 (Janvier 1963).

Les symboles de base peuvent être définis syntaxiquement au moyen des règles de grammaire suivantes :

- identificateur :

< ident > ::= letter | < ident > letter | < ident > digit

avec letter une abréviation pour :

a, b, c..... z

A, B, ... Z

avec digit une abréviation pour :

1, ... 9, 0

- entier :

< entier > ::= digit | < entier > digit

- délimiteur double :

< del double > ::= < : > =

< : > ::= :

- délimiteur simple :

< del simple > ::= delimiter | = | : |) | (

avec délimiter une abréviation pour :

+, −, ×, −, /, ↑, <, ≤, ≥, >, ≠, ≡, ∨, ∧, ¬, ⊃, 10, ;, ,, (,), [,]

(nous n'incluons pas : , ni = car ils peuvent former un délimiteur double; de même) et (caractérisent les strings);

- string :

< string > ::= < open string >)

< open string > ::= < open string > délimiteur |

< open sring > letter | < open string >

digit | < openstring > :

< open string > = | (

Ces règles de grammaire qui définissent les symboles de base d'Algol 60 sont toutes régulières. En nous remémorant l'application I, nous voyons que nous pouvons facilement construire une table séquentielle, qui reconnaît ces symboles et qui ne décrit rien d'autre que le scanner.

PHASES	Décisions	caractère suivant							
		letter	digit	delimiter	:	=	(	)	different
Start		ident	entier	del simple	:	del simple	string	del simple	erreur
Ident		ident	ident	outid	outid	outid	outid	outid	outid
Outid	"Identificateur"								
Entier		outent	entier	outent	outent	outent	outent	outent	outent
Outent	"entier"								
Del simple		outds	outds	outds	outds	outds	outds	outds	outds
Outds	"délimiteur simple "								
:		outds	outds	outds	outds	del double	outds	outds	outds
Del double		outdd	outdd	outdd	outdd	outdd	outdd	outdd	outdd
Outdd	"délimiteur double"								
String		string	string	string	string	string	string	outst	string
Outst	"string"								
Erreur	"Erreur"								

Cette table séquentielle donne la ligne générale de l'analyse; il serait intéressant d'y ajouter des instructions du niveau des différentes phases, afin de tenir compte de la construction des symboles. De plus, le scanner a aussi pour mission de supprimer les commentaires.
Nous définissons les routines et variables suivantes :

CHAR : variable globale qui contient le caractère du programme source, scanné;

A : variable de type string qui contient les caractères successifs qui constituent un symbole;

GETCHAR (GC) : routine qui scanne le caractère suivant du programme source et le place dans CHAR;

GETNOBLANK (GNB) : routine qui teste si CHAR est blanc; si oui, elle appelle GETCHAR et ce jusqu'à ce que CHAR contienne un caractère autre que blanc;

ainsi que les instructions suivantes :

ADD : ajouter à la chaîne de caractères contenus dans A, le caractère supplémentaire CHAR;

OUT (C, D) : retourner les valeurs C et D et rendre la main à l'analyseur syntaxique; C est la représentation interne du symbole scanné et D le symbole lui-même;

LOOK : vérifier si le contenu de A ne figure pas dans la table des mots-clé et délimiteurs simples; si oui, son index est à placer dans la variable J; sinon J vaut 0.
En particulier, J = 1 pour le mot-clé COMMENT.

La table complètée est T-II.

On remarquera que les commentaires sont abandonnés (sachant qu'un commentaire peut s'insérer entre le mot-clé COMMENT et un " ; " , nous n'avons considéré que ce cas là car il est le plus courant, l'autre cas (commentaire inséré entre le mot-clé END et END, ELSE ou " ; ") n'est pas difficile à introduire dans la table; nous ne l'avons pas fait pour ne pas alourdir la table inutilement).

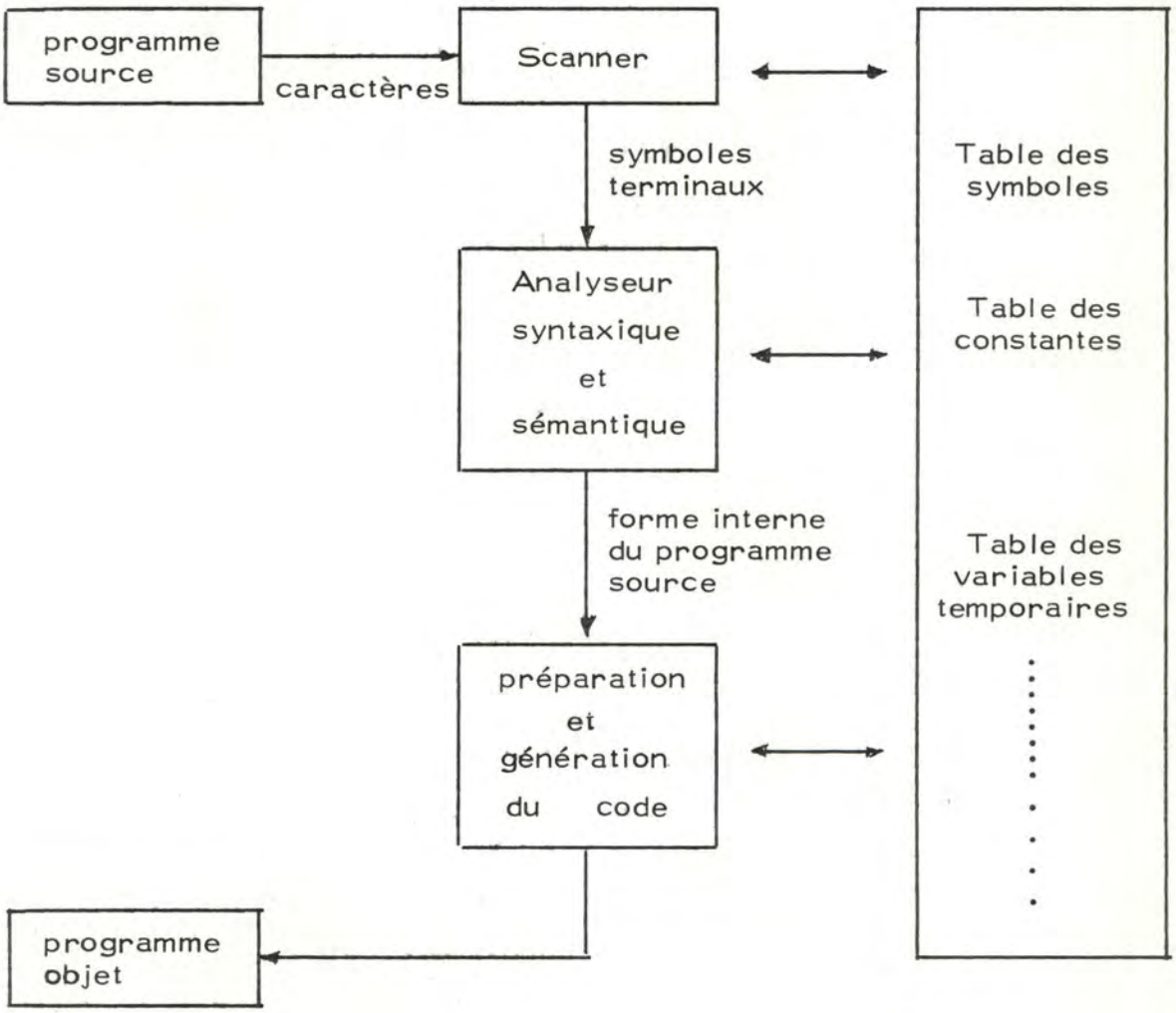
D'autre part, lorsqu'il y a erreur, on ne rend pas la main à l'analyseur syntaxique mais on scanne le symbole suivant et en sortant de la routine SCAN, CHAR contient toujours le caractère suivant du programme source.

PHASES	DECISIONS	BI	CHAR									Y		
			letter	digit	délimi-ter	:	=	(	)	;	différent	0	1	≠
Start	GNB;A vide		ident	entier	del simple	:	del simple	string	del simple	del simple	erreur			
Ident	ADD; GC		ident	ident	outid	outid	outid	outid	outid	outid	outid			
Outid	LOOK											outident	comm	outsur
Outident	OUT (§ ident, A)													
Outsur	OUT (J , A)													
Com	GC		com	com	com	com	com	com	com	com	com			
Out com		start												
Entier	ADD ; GC		outent	entier	outent	outent	outent	outent	outent	outent	outent			
Outent	OUT (§ entier, A)													
:	ADD ; GC		outds	outds	outds	outds	del double	outds	outds	outds	outds			
Del double	ADD ; GC	outdl												
Outdl	OUT (§:= , A)													
Del simple	ADD ; GC	outds												
Outds	LOOK;OUT(J, A)													
String	ADD ; GC		string	string	string	string	string	string	out-string	string	string			
Outstring	OUT (§string, A)													
Erreur	GC													

APPLICATION III - Analyse syntaxique et sémantique des programmes BASIC.

Nous avertissons le lecteur que notre but n'est pas de réaliser un analyseur syntaxique et sémantique du langage BASIC mais de montrer quel outil constituent les tables séquentielles pour mettre en page une telle réalisation.

L'analyseur syntaxique et sémantique est un module particulier des compilateurs. Il s'y insère logiquement de la manière suivante :



Le rôle de ce module est de :

- reconnaître les parties constituantes du programme;
- construire sa forme interne;
- placer les informations voulues dans les tables.

Les deux parties constituantes du module : analyseur syntaxique et analyseur sémantique travaillent de manière aussi séparée que possible : lorsque l'analyseur syntaxique a reconnu une construction du langage, il appelle une routine sémantique qui en vérifie la sémantique et place les informations nécessaires à son sujet dans la table des symboles ou dans la forme interne du programme.

1. L'analyseur syntaxique.

Nous avons choisi la définition syntaxique du Dartmouth Basic (Dartmouth Collège, 1965) dont les règles sont spécifiées en annexe.

Comme chaque instruction d'un programme écrit dans ce langage commencé par un mot clé qui la caractérise (LET, IF, FOR ....), le problème se ramène à l'analyse syntaxique des instructions qui sera réalisée grâce à une table séquentielle par type d'instruction.

Pour chaque instruction du programme, nous décodons le premier symbole, qui suit le numéro de ligne et qui est le mot clé; celui-ci alors nous aiguille vers la table qui va analyser cette instruction.

Notre but dans cette 1ère partie sera donc d'établir les tables séquentielles des différents types d'instruction.

Remarque :

Nous établirons une table supplémentaire EXP pour l'analyse des expressions; en effet celles-ci reviennent sans cesse au niveau de toutes les instructions; donc les analyser dans une table séparée allège considérablement les autres tables (dès qu'une expression est à analyser, il suffira de brancher à la table EXP pour l'analyser et ensuite revenir à la table d'origine).

Comment élaborer ces tables à partir des règles de grammaire ?

A. Par l'intermédiaire d'arbres (à raison d'un arbre par type d'instruction).

Le but de ces arbres est de déterminer aisément toutes les possibilités pour une instruction donnée, c'est-à-dire toutes les chaînes de symboles terminaux qui peuvent constituer une instruction correcte.

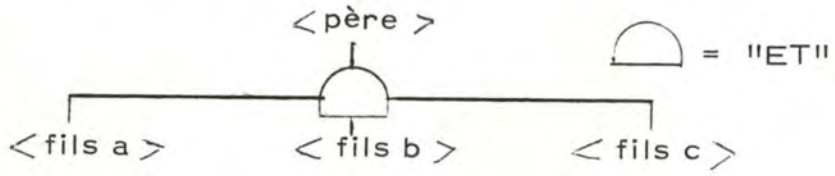
Soit une instruction particulière :

1. la racine de l'arbre est le symbole non terminal qui la représente;
2. chaque non terminal de l'arbre admet pour fils les symboles qui figurent en partie droite de la règle de grammaire dont il est partie gauche.

   a. si les symboles sont concaténés

   < père > := < fils a > < fils b > < fils c >

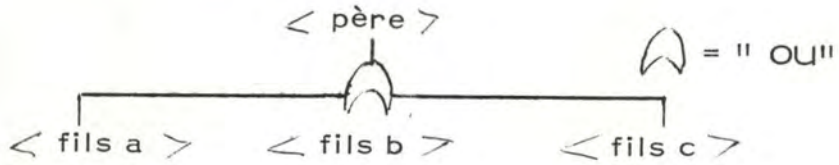
   ils sont représentés dans l'arbre par



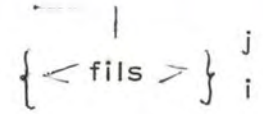
   b. si ces symboles sont reliés par " | " (ou exclusif)

   < père > := < fils a > | < fils b > | < fils c >

   ils sont représentés par



   c. si un de ces symboles figure dans la règle parenthésé par $\{ \ldots \}_i^j$ (ce qui signifie qu'il doit être répété au moins i fois et ne peut l'être plus de j fois). Cette répétition est reprise dans l'arbre par

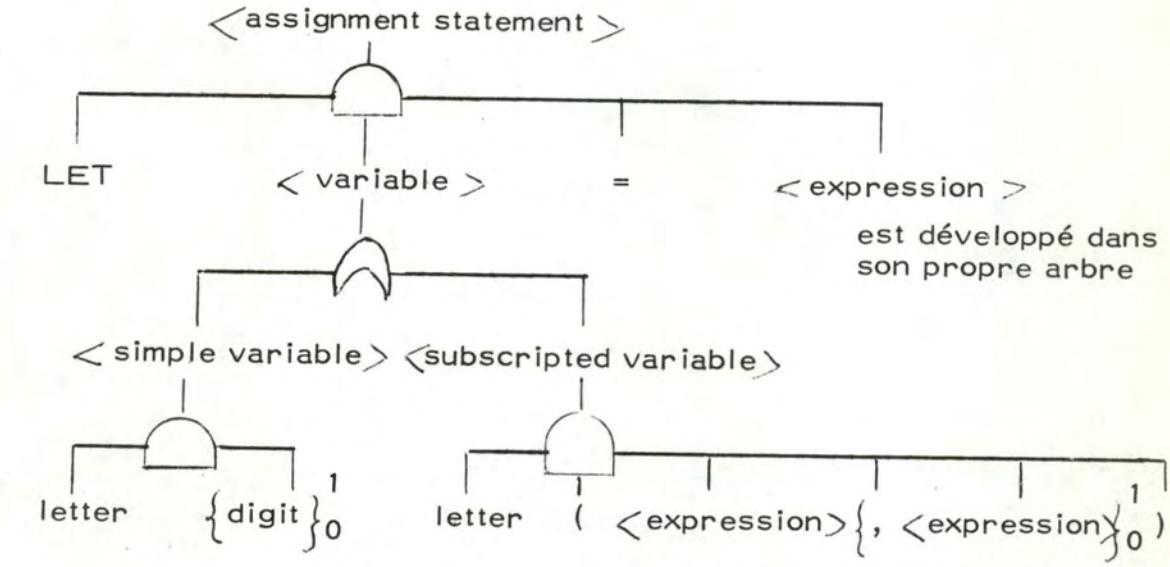


Remarque :

L'ordre gauche, droite est respecté dans l'arbre comme dans la règle.

Construisons les arbres de différents types d'instruction :

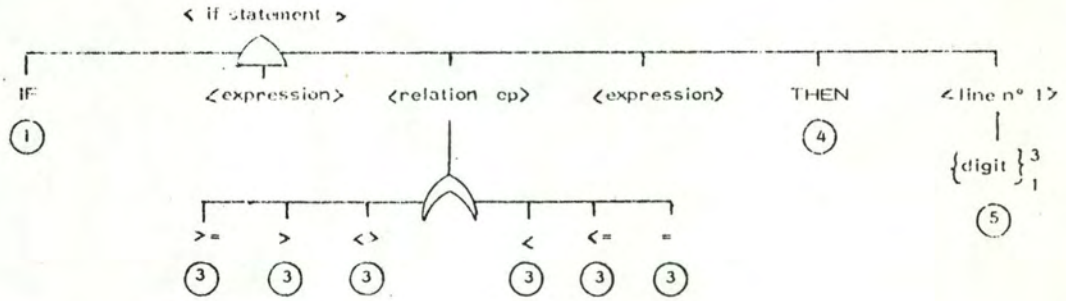
assignment statement



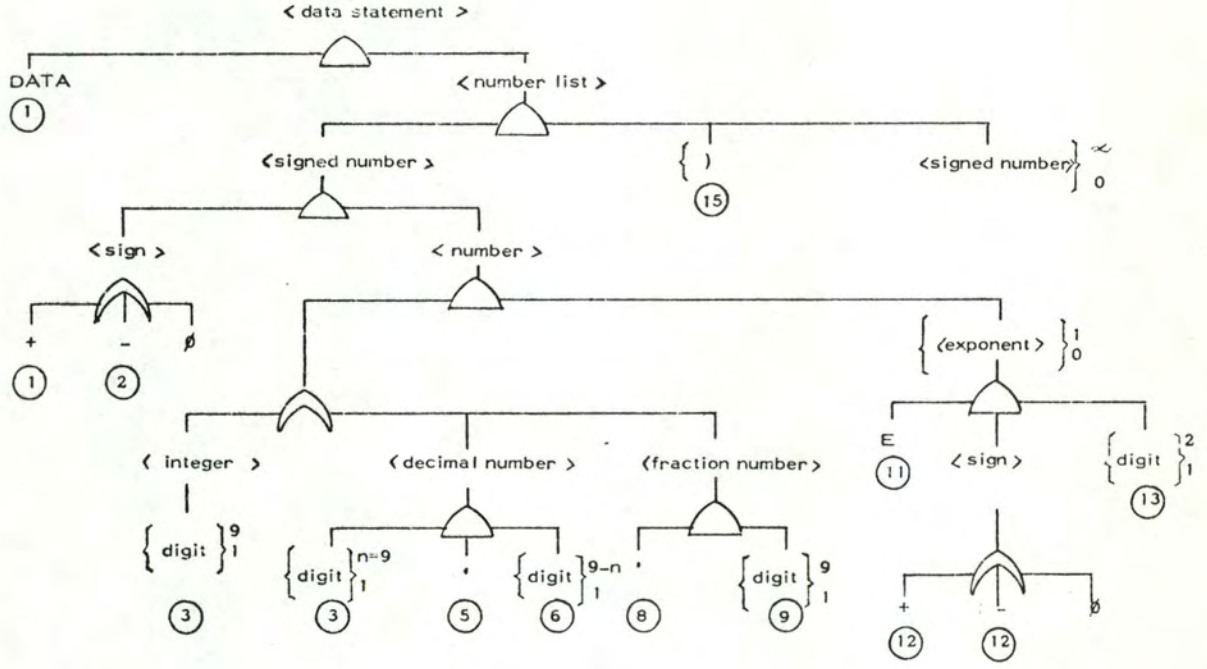
Remarque.

Nous considérons "letter" et "digit" comme symboles terminaux; ceci ne change rien et permet d'alléger l'arbre.

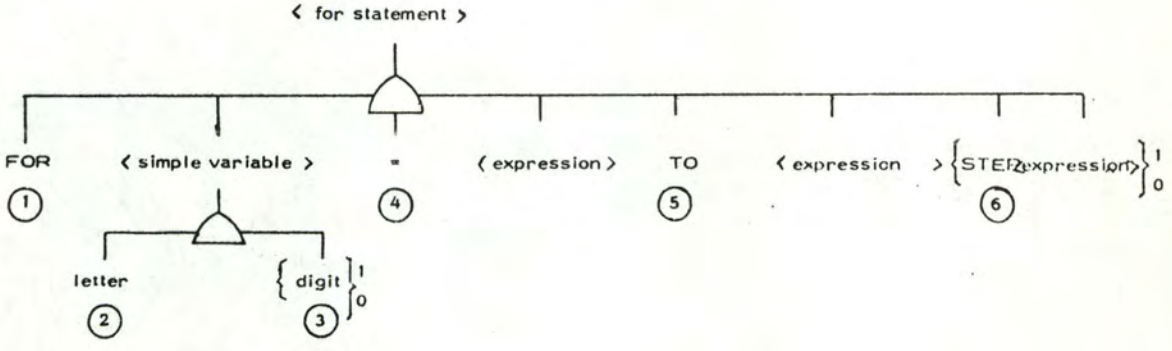
IF statement



data statement



for statement

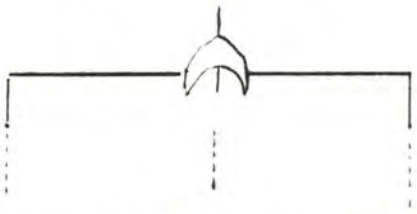


Les mêmes arbres peuvent être construits pour les autres types d'instructions :

- next statement
- dimension statement
- read statement
- print statement
- gosub statement
- return statement
- define statement.

expression

< expression >



Ici nous sommes obligés de faire une parenthèse. Dans la suite du problème, lors du passage de l'arbre à la table, nous avons constaté que par suite de la récursivité gauche qui apparaissait dans les règles de grammaire définissant la syntaxe des expression, il n'était pas possible de se servir d'une table séquentielle pour analyser une expression en raison même de la manière dont elle procède (analyse top-down de gauche à droite). En effet, une table analyse uneinstruction symbole par symbole.

Exemple :

LET < variable > = < expression >

La table vérifie si l'instruction d'assignation qu'on lui soumet est composée d'abord de LET, puis d'une variable, puis du signe = et enfin d'une expression (le même raisonnement s'appliquant à la variable et l'expression).

En ce qui concerne les expressions, pour vérifier si nous avons, par exemple, un < multiply factor > , nous sommes amené à tester recursivement que nous avons un <multiply factor > ?!

C'est pourquoi nous avons redéfini la syntaxe des expressions au moyen des règles suivantes :

$$\langle \text{expression} \rangle := \langle \text{sign} \rangle \langle \text{expression} \rangle \mid \langle \text{multiply factor} \rangle \left\{ \left\{ + \mid - \right\}_{1}^{1} \langle \text{multiply factor} \rangle \right\}_{0}^{\alpha}$$

$$\langle \text{multiply factor} \rangle := \langle \text{involution factor} \rangle \left\{ \left\{ * \mid / \right\}_{1}^{1} \langle \text{involution factor} \rangle \right\}_{0}^{\alpha}$$

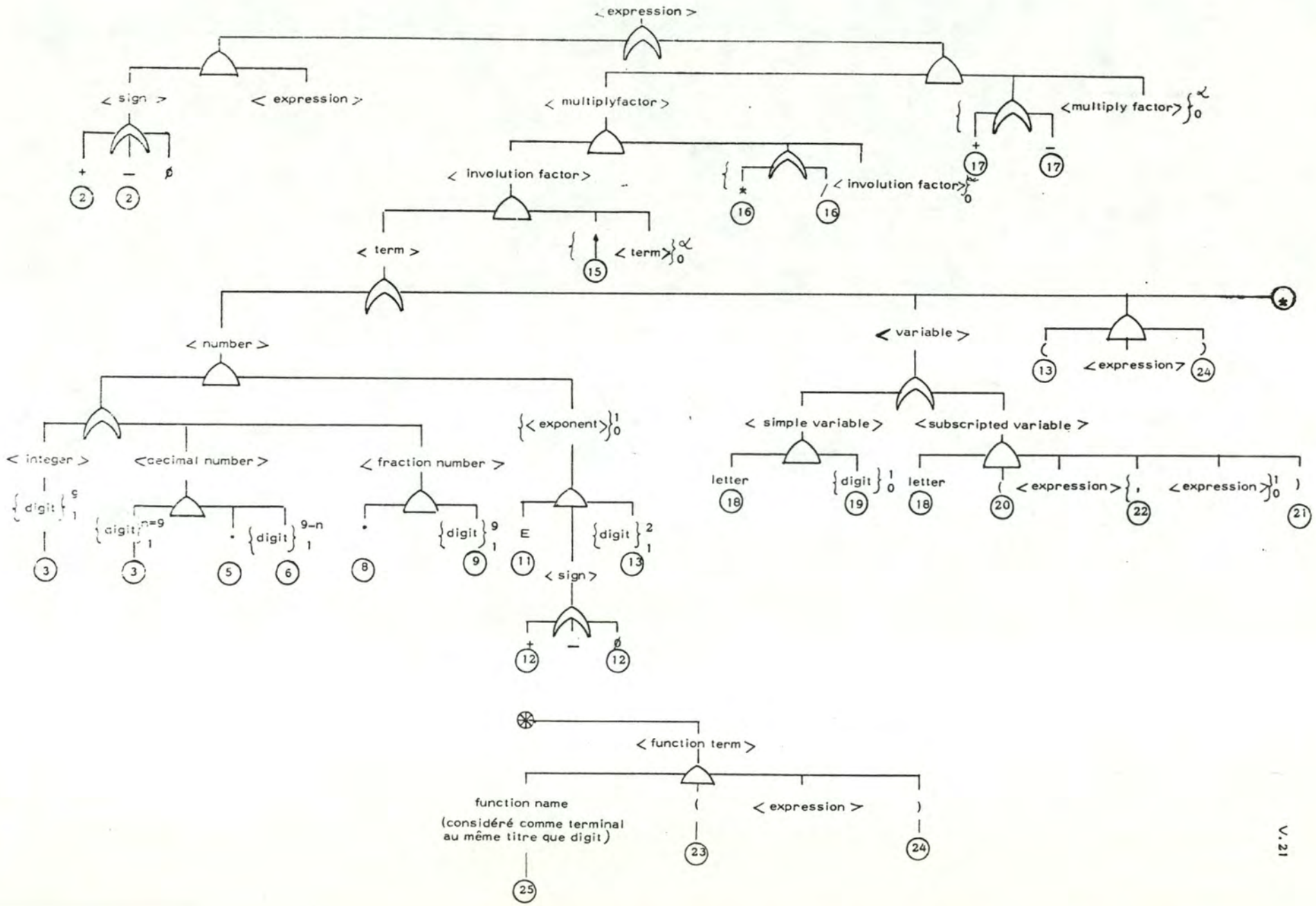
$$\langle \text{involution factor} \rangle := \langle \text{term} \rangle \left\{ \uparrow \langle \text{term} \rangle \right\}_{0}^{\alpha}$$

$$\langle \text{term} \rangle := \ldots\ldots \text{ (idem)}$$

Remarque :

- le langage engendré par l'ensemble des règles avant ou après modification, avec < BASIC program > comme symbole distingué, est le même.
- une différence est cependant à signaler : les règles comme elles ont été modifiées ne rendent plus compte de la règle d'exécution gauche, droite pour les opérateurs de même priorité. Nous verrons que c'est sans importance puisque l'analyse se fait top-down de gauche à droite.

Construisons à présent l'arbre pour les expressions.


< expression >
< sign >
< expression >
+
−
ø
2
2
< multiplyfactor >
< involution factor >
< term >
15
< term >
16
16
< involution factor >
17
17
< multiply factor >
< number >
< integer >
< decimal number >
< fraction number >
< exponent >
digit
3
3
5
6
8
9
E
11
13
< sign >
12
12
< variable >
< simple variable >
< subscripted variable >
letter
18
19
letter
18
20
22
21
13
24
function name
(considéré comme terminal
au même titre que digit)
25
< function term >
23
24

B. Elaboration des tables à partir des arbres.

Avant tout quelques considérations générales :

a. Pour chaque instruction à analyser, nous branchons à une table TAB qui après décodage du mot-clé de l'instruction aiguille le processus d'analyse vers l'une ou l'autre table;

b. Un programma BASIC est introduit en machine ligne par ligne. Le symbole terminal qui marque la fin d'une instruction est " b "

c. Le processus d'analyse que représente une table procède par lecture de gauche à droite des symboles terminaux de l'instruction source.

Elaborons la première table TAB : elle n'est établie à partir d'aucun arbre, seulement par simple considération des règles.

$$\langle \text{ Basic statement} \rangle := \langle \text{ line number} \rangle \langle \text{ statement body} \rangle$$

$$\langle \text{ Line number} \rangle := \left\{ \langle \text{ digit} \rangle \right\}_{1}^{3} \text{ b}$$

et des mots clé des instructions.

Remarque :

Les cases blanches qui figurent en ligne, 1, 2, 3, 4, 5, 11, 12 correspondent à des cas ERREUR. Dans ce cas, le message "instruction incorrecte" est envoyé et on passe à l'analyse de l'instruction suivante.

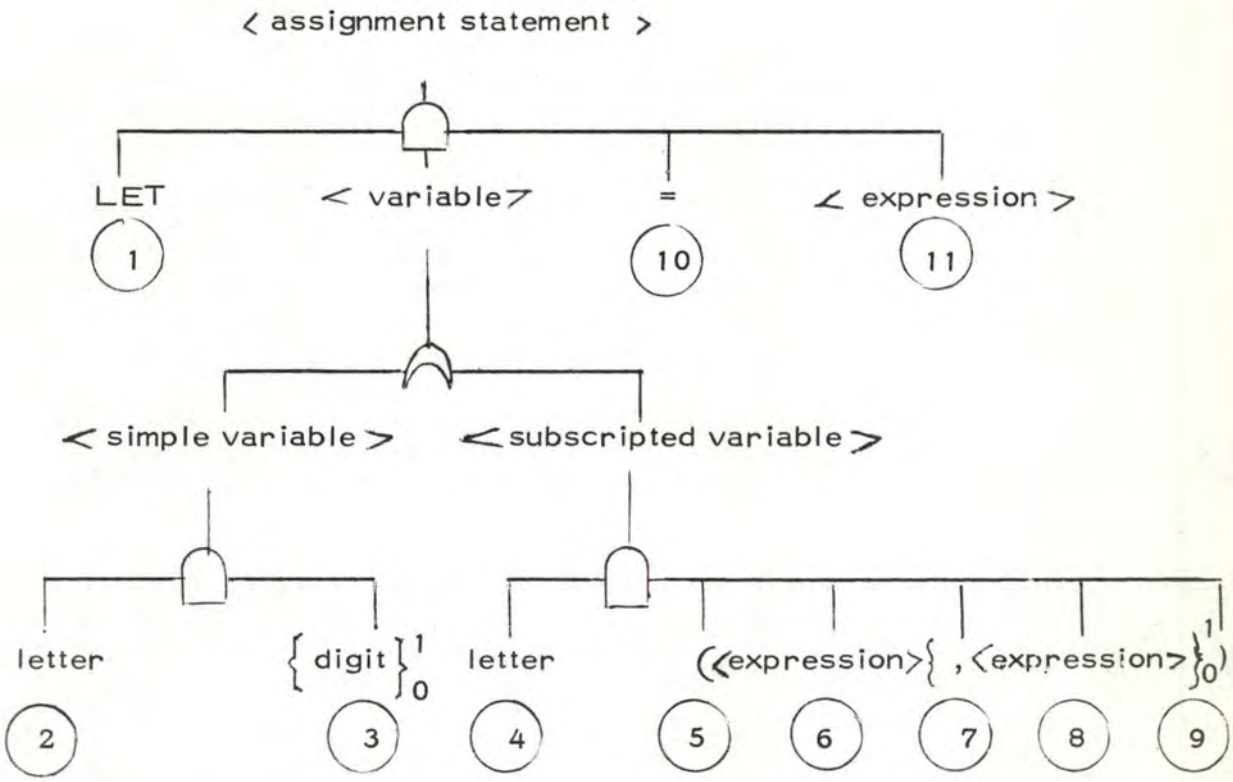
TABLE TAB

Commentaires	Phases	Décisions	CONDITIONS digit	5	END	LET	READ	DATA	PRINT	GO TO	IF	FOR	NEXT	DIM	DEF	FN	GOSUB	RETURN
	1	lire le 1er symb. terminal de l'instruction	2															
	2	lire symbole suivant	3	5														
	3	lire symbole suivant	4	5														
	4	lire symbole suivant		5														
N° de ligne	5				6	7	8	9	10	11	14	15	16	17	18		21	22
END	6	lire symbole suivant goto table END																
LET	7	lire symbole suivant goto table LET																
READ	8	lire symbole suivant goto table READ																
DATA	9	lire symbole suivant goto table DATA																
PRINT	10	lire symbole suivant goto table PRINT																
	11	lire symbole suivant		12						13								
	12	lire symbole suivant								13								
GOTO	13	lire symbole suivant go to table GOTO etc...																

Elaborons à présent les tables des différentes instructions : il suffit d'enregistrer dans ces tables toutes les séquences de symboles terminaux qui constituent une instruction correcte (2è méthode d'élaboration des tables vue en partie théorique).

Pour cette raison, attribuons à chacun des symboles terminaux figurant dans l'arbre un numéro de phase; ils nous permettront d'obtenir aisément toutes les séquences correctes.

Exemple :



(à ce niveau-ci nous considérons les expressions comme des terminaux pour rendre l'exemple plus simple).

Les séquences correctes sont : 1 , 2 , 10 , 11

1 , 2 , 3 , 10 , 11

1 , 4 , 5 , 6 , 9 , 10 , 11

1 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11

Remarques :

1. Nous pouvons, si c'est possible à ce stade-ci, fusionner des phases équivalentes c'est-à-dire intuitivement des phases qui confondues ne modifient en rien la suite de l'analyse,

   ainsi 2 ≡ 4

2. Les phases attribuées aux < expression > ne sont pas introduites dans la table; c'est au cours des phases qui les précède qu'on demande d'exécuter la table EXP, qui analyse les expressions séparément (dans l'exemple : en phase 5, 7, 10).

Table LET

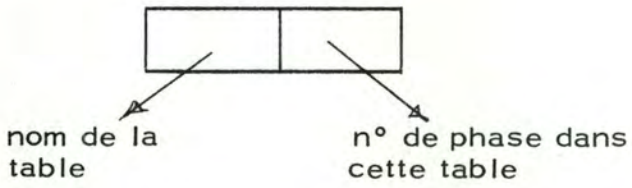
Commentaires	Phases	Décisions	letter	digit	(	)	=	,	b	≠
LET	1		2							
	2			3	5		10			
LET ⟨simple variable⟩	3						10			
	5	Do table EXP				9		7		
	7	Do table EXP				9				
LET ⟨subscr.variable⟩	9						10			
LET ⟨variable⟩ =	10	Do table EXP								FIN

Remarques :

1. En chaque phase, sauf en phase 1, la première décision à prendre est de lire le symbole terminal suivant (c'est-à-dire que l'analyseur syntaxique appelle le scanner afin qu'il le lui transmette). Nous ne l'avons pas noté dans la table pour ne pas l'alourdir .
2. Toutes les cases vides sont des cas ERREUR autrement dit l'instruction analysée n'est pas syntaxiquement correcte. Dans ce cas, on se branche à une routine d'erreur qui indique l'erreur et scanne l'instruction jusqu'au "b" pour se resynchroniser dans la table TAB.

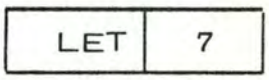
3. Interprétation du DO :

Il signifie brancher à la table indiquée en mémorisant le point de retour dans la table actuelle. Comment mémoriser ce point ? Grâce à un stack LIF 0 . Pourquoi un stack ? car la table EXP peut s'appeler recursivement.
Chaque entrée dans le stack est composée de 2 parties :



Exemple :

En supposant qu'on soit en phase 7 et qu'une expression soit à analyser, on branche à la table EXP en mémorisant :



La table EXP étant la seule qui puisse être exécutée à partir d'une autre ou d'elle-même, elle est la seule à posséder la phase RETURN.

Interprétation du RETURN : intuitivement retourner à la table origine.

Il s'agit de brancher à la table et phase spécifiées dans la dernière entrée du stack; enlever cette entrée.
Attention, en revenant à la phase abandonnée, il n'est plus nécessaire d'exécuter les décisions, il faut passer au test du symbole terminal suivant.

Le stack est initialisé (vidé) à chaque analyse d'une nouvelle instruction.

4. Ces 3 premières remarques sont valables pour toutes les tables à venir.

Table IF

PHASES	DECISIONS	>=	>	<>	<	<=	=	THEN	digit	b	≠
1	Do table EXP	3	3	3	3	3	3				
3	Do table EXP							4			
4									5		
5									6	FIN	
6									7	FIN	
7										FIN	

Table FOR

PHASES	DECISIONS	letter	digit	b	=	TO	STEP	≠
1		2						
2			3		4			
3					4			
4	Do table EXP					5		
5	Do table EXP						6	
6	Do table EXP			FIN				

Table DATA

En examinant l'arbre de <DATA statement> on remarque, par exemple, qu'un entier peut être composé de 1 à 9 digits, il faudra par conséquent introduire deux compteurs (N1 pour les parties entières, N2 pour les parties décimales) et des phases supplémentaires pour les tester.

Table DATA

Commentaires	Phases	Décisions	Bl.	+	-	b	digit	.	E	,	≠	$N_1 = 9$ oui	$N_1 = 9$ non	$N_2 = 9 - N_1$ oui	$N_2 = 9 - N_1$ non
	1	Mettre N_1 et N_2 à zéro		2	2		3	8							
⟨sign⟩	2						3	8							
⟨sign⟩ digit ...	3	Ajouter "1" à N_1										4'	4		
	4					FIN	3	5	"	15					
	4'					FIN		5	"	15					
⟨sign⟩ digit	5						6								
⟨sign⟩ digit ... . digit	6	Ajouter "1" à N_2												7'	7
	7					FIN	6		"	15					
	7'					FIN			"	15					
⟨sign⟩	8						9								
⟨sign⟩ . digit	9	Ajouter "1" à N_1										10'	10		
	10					FIN	9		"	15					
	10'					FIN			"	15					
⟨integrer⟩ ⟨dec. number⟩ E ⟨fraction⟩	11			12	12		13								
	12						13								
	13					FIN	14			15					
	14					FIN				15					
⟨signed number⟩ ,	15		1												

Remarque.

En chaque phase, il faut appeler le scanner pour connaître le symbole terminal suivant sauf en phases 1, 3, 6, 9.

| Table E X P | (voir page suivante).

C. Conclusion

Nous avons établi les tables séquentielles d'analyse c'est-à-dire qu'étant donnée une instruction, les tables indiquent s'il s'agit d'une instruction syntaxiquement correcte ou non. Si non, l'erreur se situe au niveau du dernier symbole terminal lu. Par ce processus, on abandonne l'instruction dès qu'uneerreur de syntaxe est rencontrée et la suite de l'instruction n'est plus examinée. Il pourrait être envisagé d'analyser l'erreurrencontrée et d'examiner malgré tout la fin de l'instruction.

Exemple :

Afin de fixer les idées, considérons l'analyse de l'instruction

IF A ↑ 2 + B ↑ 2 = C2 * (A4 + A6) THEN 25 ♭

Cette instruction est présentée à TAB. La lecture du 1er symbole IF nous donne la case vide (1, IF), il y a donc erreur de syntaxe (en effet cette instruction n'a pas de "line number ").

Soit 50 ♭ IF A↑ 2 + B ↑2 = C2 * (A4 + A6)THEN 25 ♭

L'analyse de cette instruction va générer la séquence des phases suivantes :

Table TAB : 5 0 ♭ I F
1 2 3 5 14 on est switché à la table IF

Table IF : 1 appel de la table EXP avec mémorisation dans le stack :

IF	1

Table E X P

Commentaires	Phases	Décisions	+	-	b	digit	.	E	letter	↑	*	/	(	)	,	function name	≠	$N_1 = 9$ oui	$N_1 = 9$ non	$N_2 = 9 - N_1$ oui	$N_2 = 9 - N_1$ non
	1	Mettre N_1 et N_2 à zéro	2	2		3	8		18				23			25					
±	2	Do table EXP			RETURN																
digit	3	Ajouter "1" à N_1																4'	4		
digit ...	4		17	17	RETURN	3	5	11		15	16	16		RETURN	RETURN		RETURN				
	4'		17	17	RETURN		5	11		15	16	16		RETURN	RETURN		RETURN				
	5					6															
	6	Ajouter "1" à N_2																		7'	7
digit.... digit .	7		17	17	RETURN	6		11		15	16	16		RETURN	RETURN		RETURN				
	7'		17	17	RETURN			11		15	16	16		RETURN	RETURN		RETURN				
	8					9															
	9	Ajouter "1" à N_1																10'	10		
. digit ...	10		17	17	RETURN	9		11		15	16	16		RETURN	RETURN		RETURN				
	10'		17	17	RETURN			11		15	16	16		RETURN	RETURN		RETURN				
⟨intégrer⟩ ⟨fraction⟩ E ⟨dec. number⟩	11		12	12		13															
idem E ±	12					13															
	13		17	17	RETURN	14				15	16	16		RETURN	RETURN		RETURN				
⟨number⟩	14		17	17	RETURN					15	16	16		RETURN	RETURN		RETURN				
	15					3	8		18				23			25					
	16					3	8		18				23			25					
	17					3	8		18				23			25					
letter	18		17	17	RETURN	19				15	16	16	20	RETURN	RETURN		RETURN				
⟨simple variable⟩	19		17	17	RETURN					15	16	16		RETURN	RETURN		RETURN				
letter (	20	Do table EXP												21	22						
⟨subscr. variable⟩	21		17	17	RETURN					15	16	16		RETURN	RETURN		RETURN				
	22	Do table EXP												21							
	23	Do table EXP												24							
⟨function term⟩ ⟨expression⟩)	24		17	17	RETURN					15	16	16		RETURN	RETURN		RETURN				
function name	25												23								

Table EXP : A ↑ 2 + B ↑ 2 =

1 18 15 3 17 18 15 3 RETURN

on rebranche à la table IF, en phase 1. Le stack est à nouveau vide.

Table IF (1) : =

3

appel de la table EXP avec mémorisation dans le stack.

Table EXP : C 2 * (

1 18 19 16 23

appel de la table EXP avec mémorisation dans le stack.

EXP	23
IF	3

Table EXP : A 4 + A 6)

RETURN : phase 23, table EXP.

Table EXP :) THEN

(23) 24 RETURN : phase 3, table IF

Table IF : THEN 2 5 6

(3) 4 5 6 (FIN)

2. L'analyseur sémantique. *

Pour la facilité et la clarté de l'application, nous avons séparé en première partie les deux analyseurs, pour nous consacrer uniquement à l'analyseur syntaxique. Dans cette deuxième partie, nous n'allons pas en réalité ne considérer que l'analyseur sémantique mais voir comment et quand l'analyseur syntaxique appelle l'analyseur sémantique et en quoi consiste l'intervention de celui-ci. Nous reprendrons les tables établies en partie 1 et nous les enrichirons au niveau décision de routines sémantiques.

A. Généralités.

Forme interne d'un programme.

Une des fonctions des routines sémantiques est la génération de la forme interne du programme.

Quelques mots concernant cette forme interne : elle peut prendre plusieurs formes possibles. L'une d'elles est une liste de quadruples (opérateur, opérande 1, opérande 2, résultat) dans l'ordre dans lequel ils doivent être exécutés. Cette forme de quadruples est bien adaptée aux opérations binaires et unaires,

Exemple : l'instruction d'affectation A := B + C * D correspond à la liste

*, C, D, T_1

+, B, T_1, T_2

:=, T_2,, A (où T_1 et T_2 sont des variables temporaires crées par le compilateur)

Attention les opérandes ne sont pas des noms symboliques mais des pointeurs vers des éléments de la table des symboles.

et peut être étendue aux autres opérateurs :

Opérateur	Opérandes	Interprétation
BR	i	brancher au quadruple i
B Z / P / M	i, P	brancher au quadruple i si la valeur de l'élément pointé par P est nulle, positive, négative

---

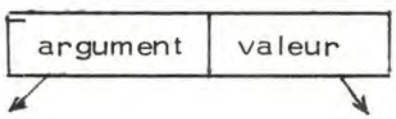
* Adaptation au langage BASIC de l'analyseur sémantique développé par D. GRIES dans son ouvrage "Compiler construction for digital computers".

+ * / −	P_1, P_2, P_3	additionner , multiplier, diviser, soustraire la valeur de l'élément pointé par P_1 à celle de l'élément pointé par P_2; placer le résultat dans l'élément pointé par P_3.
:=	P_1, , P_3	affecter la valeur pointée par P_1 à l'élément pointé par P_3
B G L E	i, P_1, P_2	brancher au quadruple i si la valeur pointée par P_1 est $>$, $<$, = à celle pointée par P_2;
BRL	P	brancher au quadruple décrit par l'élément pointé par P.

Tables d'information.

Pour la facilité de l'exposé, nous supposerons que table des symboles (qui contient tous les identificateurs figurant dans le programme source, avec leurs attributs), table des constantes, table des variables temporaires créées par le compilateur, sont fusionnées en une seule pour laquelle nous garderons la dénomination : table des symboles (notée TS).

Une entrée dans cette table possède deux parties constituantes



identifie l'élément

- constante
- tableau
- variable
  - simple
  - indicée
  - temporaire
- fonction

décrit tous les attributs de l'élément

comme attributs d'un élément signalons :

- TYPE :
  - = 1 variable simple
  - = 2 tableau
  - = 3 variable indicée
  - = 4 fonction
  - = 5 constante

- TEMPORARY = 0 variable non temporaire
  = 1 variable temporaire

- NUMBER : si TYPE = 2 donne le nombre de dimensions du tableau

  si TYPE = 3 donne l'adresse dans TS de l'élément qui décrit le tableau

  si TYPE = 4 donne le n° du 1er quadruple correspondant à la définition de la fonction

- ADDRESS : si TYPE = 3 adresse de la partie variable qui additionnée à la partie constante permettra de trouver l'élément désiré du tableau ; en effet rappelons qu'étant donné A [n, n] :

  adresse de A [i , j] =

  $$\underbrace{(\text{adresse } A[1,1] - n - 1)}_{\text{partie constante}} + \underbrace{(i * n + j)}_{\text{partie variable}}$$

  si TYPE = 4 adresse de l'expression argument de la fonction

  si TYPE = 5 numéro du 1er quadruple correspondant à l'instruction que cette constante identifie (dont elle est < line number >) si tel est le cas.

Nous ne signalons que les attributs dont nous aurons besoin ; il en existe d'autre tels l'adresse de l'élément dans le programme objet, les domaines des indices si l'élément est un tableau .....

Les routines sémantiques travaillent en permanence avec cette table, soit pour y insérer un nouvel élément, soit pour vérifier qu'un élément y figure, soit pour y prélever des informations. Elle utilise à cet effet les 2 procédures :

LOOKUP (NAME, P) recherche dans la table l'entrée d'argument NAME et place l'adresse de cette entrée dans la variable P. Si elle n'existe pas, P est positionné à 0.

INSERT (NAME, P) insère un nouvel élément identifié par NAME et place son adresse dans P.

Elles référencent les attributs d'un élément dont l'adresse dans la table est donnée par P, par P. TYPE, P. ADDRESS etc. . .

Routines sémantiques.

Elles sont appelées par l'analyseur syntaxique lorsque celui-ci a reconnu une construction. Elles analysent la sémantique de la construction, génèrent la forme interne correspondante sous forme de quadruples, introduisent si nécessaire, de nouvelles informations dans la table des symboles. A cet effet, elles utiliseront les routines et variables (globales) suivantes :

Routines :

ENTER (P_0 , P_1, P_2, P_3) génère un quadruple dont les éléments sont les 4 arguments

P_0 : opérateur

P_1, P_2, P_3 : pointeurs vers des éléments de la table des symboles - (TS) .

GENERATEMP (P) : insère un nouvel élément dans TS pour une variable temporaire et place son adresse dans P.

ADD (X) : ajoute (par concaténation) au contenu de X, le contenu de NEXTSYMB (qui contient toujours le symbole terminal suivant du programme source).

CONVERTRI (P) : vérifie si l'élément décrit par P est entier si non, le convertit et appelle GENERATEMP afin de créer une nouvelle variable pour y placer le résultat de la conversion.

Variables :

NEXTQUAD : variable qui contient toujours le numéro du quadruple suivant;

QUAD (I, J) : référence la jème composante du quadruple i .

B. Routines sémantiques de traitement des variables indicées.

Avant de reprendre les tables séquentielles pour y introduire les routines sémantiques, nous allons expliciter les routines sémantiques qu'appelle l'analyseur syntaxique lorsqu'il a reconnu une variable indicée, peu importe la table dans laquelle il est en train d'opérer.

Ces routines doivent générer le code (quadruples) nécessaire à la détermination des indices et le code correspondent aux calculs de détermination de l'adresse de l'élément de tableau considéré, au moment de l'exécution.
Elles procèdent sur les stacks COUNT, ARR, ENTRY, ENTRY2, dont les entrées contiennent respectivement :

- le nombre de dimension du tableau
- le nom du tableau
- l'adresse des éléments décrivant le domaine des indices
- l'adresse de la variable temporaire pour la partie variable du calcul de l'adresse.

Soient leur pointeur respecitf : CP - AP - EP - EP2, initialement positionnés à 1.

Pourquoi des stacks ? parce que ces routines traitent les variables indicées et que leurs indices peuvent eux-mêmes être des variables indicées (problème de récursivité).

Lorsque l'analyseur syntaxique a reconnu

letter (< expression > .....

où letter est l'identificateur du tableau, placé dans NAME

la routine sémantique appelée est la suivante : SEM 1 (écrite en "Bastard Algol")

```
LOOKUP (NAME, P) ;
IF P = 0 OR P.TYPE ≠ 2 THEN ERROR;
CP := CP + 1 ;
COUNT [CP] := P.NUMBER - 1 ;
AP := AP + 1 ;
ARR [AP] := P ;
EP := EP + 1 ;
ENTRY[ EP] := P + 1 ;   l'élément suivant de TS décrit
                        le domaine du 1er indice.
```

```
GENERATEMP (P) ; génération d'une variable temporaire
                 pour la partie variable du calcul de
                 l'adresse.
EP2 := EP2 + 1 ;
ENTRY2 [EP2] := P;
CONVERTRI (X1) ; l'adresse de la variable temporaire
                 qui contient le résultat de l'expression
                 est pointée par X1.
ENTER (:= , X1 , 0 , ENTRY2 [EP2] :
```

après avoir reconnu, s'il existe, le 2ème indice

..... , < expression > .....

il faut exécuter SEM 2 :

```
COUNT [CP] := COUNT [CP] - 1;
ENTRY [EP] := ENTRY [EP] + 1 ; l'élément pointé
                               donne le domaine du 2ème
                               indice;
ENTER (* , ENTRY2 [EP2] , ENTRY [EP] , ENTRY2
                                        [EP2]) ;
CONVERTRI (X1) ;
ENTER (+ , ENTRY 2 [EP2] , X1, ENTRY2 [EP2] ;
```

après avoir reconnu la parenthèse fermée ....), exécution de SEM3 :

```
IF COUNT [CP] ≠ 0 THEN ERROR ;
GENERATEMP (P) ;  génération d'un élément dans TS
                  pour la variable indicée;
P. TYPE := 3 ;
P. ADDRESS := ENTRY2 [EP2] ;   } voir la signification
P. NUMBER := ARR [AP] ;        } des attributs
CP := CP - 1 ;
AP := AP - 1 ;
EP := EP - 1 ;
EP2 : = EP2 - 1 ;
```

C. Routines sémantiques associées aux expressions.

Au cours de l'analyse syntaxique d'une expression, les routines sémantiques appelées doivent :

- vérifier si les opérandes figurent dans TS;
- générer les quadruples nécessaires.

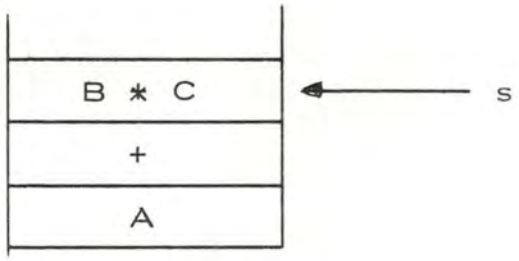
Afin d'alléger la table EXP, nous l'éclaterons en 4 tables :

- table EXP (< expression >)
- table M. F. (< multiply factor >)
- table I. F. (< involution factor >)
- table TERM (< term >)

Les routines appelées par ces différentes tables procèdent sur un stack LIFO "ST" dont les entrées sont pointées par "s". "ST" contient à tout moment les adresses des opérandes (éventuellement résultat) de l'expression ainsi que les opérateurs qui les lient; ceci, pour la partie de l'expression déjà analysée.

Exemple : A + B * C - D
symbole suivant à analyser

"ST" contient :



A ce stacksont associées 2 routines :

INPUTST (X) qui a pour effet { s := s + 1; ST [s] := X

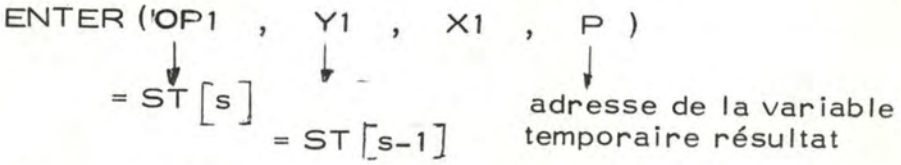
variable

OUTPUTST (X) qui a pour effet { X := ST [s] ; s := s - 1

Les routines appelées par les différentes tables procèdent également sur les variables globales : OP1, X1, Y1 qui contiennent toujours avant génération d'un quadruple, l'opérateur qui est associé à ce quadruple et l'adresse de ses 2 opérandes.

Le principe général est le suivant :

- tout opérateur rencontré est introduit dans ST
- lors de la recherche d'un des 2 opérandes d'un opérateur, après appel des tables correspondantes (EXP, M. F., I. F., TERM), on revient toujours avec son adresse dans la variable X1;
  - . si l'opérande est 1er opérande de l'opérateur, son adresse est introduite dans ST;
  - . si l'opérande est 2ème opérande de l'opérateur, il faut générer le quadruple déterminé par cet opérateur;



puis affecter l'adresse du résultat à X1 et diminuer le stack ST de 2 entrées.

Reprenons à présent les tables d'analyse syntaxique des expressions en les enrichissant au niveau décision des instructions sémantiques requises.

Table EXP

COMMENTAIRES	PHASES	TRAITEMENTS	NEXTSYMB +	NEXTSYMB -	NEXTSYMB ≠
	1		2	2	3
	2	INPUTST (NEXTSYMB); SCAN;	Return	Return	Return
SCAN = appel du scanner qui place le symbole terminal suivant dans NEXTSYMB					
Recherche de l'<expression>; au retour l'adresse de l'expression sera dans la variable X1.		Do Table EXP;			
Génération d'une variable temporaire.		GENERATEMP (P); OUTPUTST (OP1);			
Génération du quadruple correspondant à l'expression détectée.		ENTER (OP1, 0, X1, P);			
Mise de l'adresse de l'expression dans X1.		X1 := P;			
Rechercher l'opérande < multiply factor >	3	Do Table M. F. ;	4	4	Return
	4	INPUTST (X1) INPUTST(NEXTSYMB); SCAN ;	4	4	Return
Mise de l'opérande et de l'opérateur au sommet de ST					
Rechercher l'opérande < multiply factor >		Do Table M. F. ; GENERATEMP (P) ; OUTPUTST (OP1) ; OUTPUTST (Y1);			
Générer le quadruple correspondant à l'expression et mettre son adresse dans X1.		ENTER (OP1, Y1, X1, P); X1 := P			

Table M. F.

COMMENTAIRES	PHASES	TRAITEMENTS	NEXTSYMB *	/	≠
Rechercher l'opérande ⩽involution factor⩾ .	1	Do table I. F.	2	2	Return
Mettre l'opérande et l'opérateur au sommet du stack ST. Rechercher l'opérande ⩽ involution factor⩾. Générer le quadruple correspondant à l'opérateur et ses 2 opérandes. Placer l'adresse du résultat dans X1.	2	INPUTST (X1); INPUTST(NEXTSYMB); SCAN Do table I. F. ; GENERATEMP (P) ; OUTPUTST (OP1); OUTPUTST (Y1) ; ENTER (OP1, Y1, X1, P); X1 := P	2	2	Return

Table I.F.

COMMENTAIRES	PHASES	TRAITEMENTS	NEXTSYMB ≠
Rechercher l'opérande < term > .	1	Do table TERM;	2 Return
Mettre l'opérande et l'opérateur au sommet du stack ST. Rechercher l'opérande < term> Générer le quadruple correspondant à l'opérateur et ses 2 opérandes. Placer l'adresse du résultat dans X1.	2	INPUTST(X1) ; INPUTST(NEXTSYMB); SCAN ; Do table TERM; GENERATEMP(P); OUTPUTST (Y1) ; OUTPUTST (OP1); ENTER(OP1, Y1, X1, P); X1 := P	2 Return

Commentaires	PHASES	TRAITEMENTS	+	-	b	digit	.	E	letter	(	*	/	(	)	,	↑	≠	$N_1 = 9$ oui	$N_1 = 9$ non	$N_2 = 2 \cdot N_1$ oui	$N_2 = 2 \cdot N_1$ non
X est destiné à contenir le < term> cherché	1	X := ' ' ; N1 := 0 ; N2 := 0				3	8		18				23			25					
	3	N1:= N1 + 1 ; ADD (X)																4'	4		
	4	SCAN	15	15	15	3	5	11		15	15	15		15	15		15				
	4'	SCAN	15	15	15		5	11		15	15	15		15	15		15				
	5	ADD (X) ; SCAN				6															
	6	N2 := N2 + 1 ; ADD (X)																		7'	7
	7	SCAN	15	15	15	6		11		15	15	15		15	15		15				
	7'	SCAN	15	15	15			11		15	15	15		15	15		15				
	8	ADD(X) ; SCAN				9															
	9	N1 := N1 + 1 ; ADD (X)																10'	10		
	10	SCAN	15	15	15	9		11		15	15	15		15	15		15				
	10'	SCAN	15	15	15			11		15	15	15		15	15		15				
	11	ADD (X) ; SCAN	12	12		13															
	12	ADD (X) ; SCAN				13															
	13	ADD (X) ; SCAN	15	15	15	14				15	15	15		15	15		15				
	14	ADD (X) ; SCAN	15	15	15					15	15	15		15	15		15				
Le constante figure-t-elle dans TS ? Elle est le term cherché. Introduire son adresse dans X1.	15	LOOKUP(X, P); IF P = 0 THEN ERROR; X1 := P; RETURN																			
	18	ADD (X) ; SCAN	19'	19'	19'	19'				19'	19'	19'	20	19'	19'		19'				
	19	ADD(X) ; SCAN	19'	19'	19'					19'	19'	19'		19'	19'		19'				
La variable simple figure-t-elle dans TS ? Elle est le term cherché. Introduire son adresse dans X1.	19'	LOOKUP (X, P); IF P = 0 THEN ERROR; X1 := P; RETURN																			
Mémorisation de l'identificateur du tableau dans le stack Z. Routine sémantique associée au 1er indice.	20	INPUTZ (X); SCAN ; Do Table EXP ; OUTPUTZ(NAME); SEM 1												21	22						
	21	SEM 3 ; SCAN	22'	22'	22'					22'	22'	22'		22'	22'		22'				
Routine sémantique associée au 2ème indice.	22	SCAN Do table EXP ; SEM 2												21							
Introduire l'adresse de la variable indicée dans X1.	22'	X1 := P; RETURN																			
	23	SCAN Do table EXP ;												24							
Introduire l'adresse de la variable temporaire résultat de l'expression dans X1.	24	X1 := P; SCAN	24'	24'	24'					24'	24'	24'		24'	24'		24'				
	24'	RETURN																			
Mémorisation du nom de la fonction dans le stack Z.	25	INPUTZ (X) ; SCAN											26								
	26	SCAN ; Do table EXP OUTPUTZ (FUN) ;												27							
Mémorisation de l'adresse de retour Génération de l'affectation de l'adresse de retour à la variable Y (le dernier quadruple généré par la fonction est BRL, Y).	27	LOOKUP (FUN, P); P. ADDRESS := X1 ; ENTR := P; J := NEXTQUAD + 2; LOOKUP (J, P) ; IFP = 0 THEN INSERT (J, P ENTER (:=, P, 0, Y)	24'	24'	24'					24'	24'	24'		24'	24'		24'				

D. Routines sémantiques associées à l'instruction LET.

Table LET

Commentaires	PHASES	TRAITEMENTS	B. I.	Letter	digit	(	)	=	,	b	≠
NAME est destiné à contenir la variable affectée.	1	NAME := ' '		2							
	2	ADD (NAME) ; SCAN			3	5		10'			
La variable simple affectée est-elle dans TS ? Si non lui associer une entrée dans TS.	3	ADD (NAME) ; LOOKUP (NAME, P) ; IF P = 0 THEN BEGIN INSERT (NAME, P) ; P. TYPE := 1 ; P. TEMPORARY := 0 END ; SCAN						10			
Routine sémantique associée au 1er indice de la variable	5	SCAN ; Do table EXP ; SEM 1 ;					9		7		
Routine sémantique associée au 2ème indice de la variable.	7	SCAN ; Do table EXP ; SEM 2 ;					9				
	9	SEM 3 ; SCAN						10			
ENTRY 1 point vers l'élément associé à la variable affectée, dans TS Génération du quadruple	10	ENTRY 1 := P ; SCAN ; Do table EXP ; ENTER(:=, X1, 0, ENTRY1)								FIN	
La variable simple affectée est-elle dans TS ? Si non lui associer une entrée dans TS.	10'	LOOKUP (NAME, P) ; IF P = 0 THEN BEGIN INSERT (NAME, P) ; P. TYPE := 1 ; P. TEMPORARY := 0 END ;	10								

La sémantique introduite au niveau des décisions a pour but de :

- si la variable à affecter est une variable simple, vérifier si elle figure dans la table des symboles, sinon y introduire son identificateur et ses attributs;

- si la variable à affecter est une variable indicée, vérifier si le tableau a été déclaré, générer les quadruples nécessaires à la détermination des indices et de l'adresse de l'élément du tableau (routines sémantiques SEM_i ; i = 1, 2, 3).

- générer le quadruple d'affectation.

E. Routines sémantiques associées à l'instruction IF.

Commentaires	PHASES	TRAITEMENTS	>=	>	<>	<	<=	=	THEN	digit	b	≠
B est destinée à contenir < line n° 1 > Analyse syntaxique et sémantique de la 1ère expression Introduire son adresse dans R	1	B := ' ' ; Do table EXP ; R := X1	2	3	4	5	6	7				
Analyse syntaxique et sémantique de la 2ème expression. Générer le quadruple de branchement si la 1ère expression est ⩾ que la 2ème, à NEXTQUAD + 1	2	SCAN ; Do table EXP ; ENTER(BGE, NEXTQUAD+1, R, X1)							8			
Idem (>)	3	SCAN ; Do table EXP ; ENTER (BG, NEXTQUAD+1, R, X1)							8			
Idem (≠)	4	SCAN ; DO table EXP ; ENTER (BNE, NEXTQUAD+1, R, X1)							8			
Idem (<)	5	SCAN ; Do table EXP ; ENTER (BL, NEXTQUAD+1, R, XI)							8			
Idem (⩽)	6	SCAN ; Do table EXP ; ENTER (BLE, NEXTQUAD+1, R, X1)							8			
Idem (=)	7	SCAN ; Do table EXP ; ENTER (BE, NEXTQUAD+1, R, X1)							8			
Générer le quadruple de branchement au quadruple NEXTQUAD+1 au cas où la relation n'est pas vérifiée.	8	ENTER (BR, NEXTQUAD+1, 0, 0); SCAN								9		
	9	ADD(B) ; SCAN								10	11'	
	10	ADD(B) ; SCAN								11	11'	
Générer le quadruple de branchement au quadruple associée à< line no 1 >	11	ADD(B) ; SCAN LOOKUP (B, P) ; ENTER(BRL, P, 0, 0,)									FIN	
Idem	11'	LOOKUP (B, P) ; ENTER (BRL, P, 0, 0) FIN										

F. ROutines sémantiques associées à l'instruction FOR.

Table FOR

Commentaires	PHASES	TRAITEMENTS	letter	digit	b	=	To	STEP	≠
A est destiné à contenir la variable de boucle	1	A := ' ' ;	2						
	2	ADD(A) ; SCAN		3		4			
	3	ADD(A) ; SCAN				4			
Placer l'adresse de la variable dans TS, dans P. Si elle n'y figure pas, l'y introduire.	4	SCAN ; Do table EXP ; LOOKUP (A, P) ; IF P = 0 THEN BEGIN INSERT (A, P) ; P. TYPE:= 1 ; END ;					5		
quadruple d'affectation initiale		ENTER (:=, X1, O, P);							
Mémorisation de l'adresse		ENTRY := P ;							
Mémorisation du n° du quadruple suivant		JUMP := NEXTQUAD ;							
Quadruple de branchement au test de dépassement, mais on en connait pas le n° de quadruple associé		ENTER (BR, O, O, O); JUMP2 : = NEXTQUAD							
	5	SCAN ; Do table EXP			7			6	
Recherche de l'adresse de "1" dans TS.	7	LOOKUP (1, P) ;							
Quadruple d'incrémentation d'un pas "1"		ENTER(+, ENTRY, P, ENTRY);							
on connait à présent l'adresse de branchement (n° de quadruple = quadruple suivant)		I := JUMP ; QUAD (I, 2):=NEXTQUAD; JUMP:=NEXTQUAD ;							
quadruple de test de dépassement		ENTER (BG, O, ENTRY, X1); FIN							
	6	SCAN ; Do table EXP ;			7'				
Idem	7'	ENTER(+, ENTRY, X1, ENTRY) ; I := JUMP ; QUAD (I, 2):=NEXTQUAD; JUMP := NEXTQUAD; ENTER(BG, O, ENTRY, X1); FIN							

Remarques :

- nous avons supposé le pas positif et la valeur limite $\leq$ valeur initiale ;

- la sémantique associée à l'instruction NEXT complémentaire de FOR est :

```
ENTER (BR, JUMP2, O, O)  quadruple de branchement à
                         l'incrémentation;
I := JUMP
QUAD (I, 2) := NEXTQUAD : Introduction en bonne place du
                          quadruple suivant comme qua-
                          druple auquel on se branche dans
                          le cas où il y a dépassement.
```

G. Conclusion et exemple.

Nous n'avons pasau cours de cette application pris en considération toutes les instructions du BASIC. Cela aurait été trop long. En particulier, nous nous sommes intéressés aux instructions qui au niveau sémantique provoquaient la génération de quadruples dans la forme interne du programme.

L'analyse syntaxique des instructions que nous n'avons pas envisagé n'offre aucune difficulté, si le lecteur désirait élaborer leur table séquentielle. Quant aux routines sémantiques qui leur sont associées, comme pour les autres tables, elles sont fonction de l'instruction. Notons parmi elles, les instructions READ, DIM, pour lesquelles les routines sémantiques n'ont pas de quadruples à générer, seulement réserver les entrées nécessaires dans TS.

Terminons par un exemple plus concret, en considérant le morceau de programme :

```
10    LET K = 0
20    IF I > J THEN50
30    LET I = I + 1
40    GOTO 20
50    LET K = K + A (I-1, J * 3) * 6
60    FOR J = 1 TO K + 1
70    LET N = N + J
80    NEXT J
 .     .
 .     .
 .     .
```

Nous supposons que K, I, J, N figurent dans la table des symboles ainsi que les constantes apparaissant dans le programme.
Pour faciliter les écritures, nous désignons les pointeurs vers les éléments de TS par leur identificateur : exemple : K pointe vers l'élément de TS qui décrit la variable K; "0" point vers l'élément de TS associé à la constante "0".
Les variables temporaires créées par l'analyseur sémantique sont notées T_i (I = , 2, ...)

L'analyseur syntaxique ne détecte pas d'erreur.

L'analyseur sémantique (les routines) génère la liste des quadruples suivante :

```
(1)     := , O, , K
(2)     BG , 4 , I , J
(3)     BR 5
(4)     BR 8
(5)     + , I , 1 , T1
(6)     := , T1, , I
(7)     BR 2
(8)     - , I , 1 , T2
(9)     := , T2, , T3
(10)    * , J , 3 , T4
(11)    * , T3 , D2 , T3
(12)    = , T3 , T4 , T3
(13)    * , T5 , 6 , T6
(14)    + , K , T6 , T7
(15)    := , T7 , ,K
(16)    BR 19
(17)    +, J , 1 , J
(18)    +, K , 1 , T8
(19)    BG , 23 , J , T8
(20)    + , N , J , T9
(21)    := , T9, , N
(22)    BR 17
(23)    .
        .
        .
```

quadruples qui calculent la partie variable nécessaire à la détermination de l'adresse à l'exécution de l'élément du tableau.
D2 donne la dimension du tableau pour le 2ème indice.
La variable indicée dans TS est pointée par T5.

3. Annexe.

APPENDIX C

A Formal Definition of Dartmouth Basic†

<alphabetic character> := A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z
<digit> := 0|1|2|3|4|5|6|7|8|9
<special character> := +|−|*|/|b|=|(|)|>|<|.|,|; |↑
<integer> := $\{$<digit>$\}_1^9$
<fraction> := .<integer>
<decimal number> := $\{$<digit>$\}_1^{n\leq 9}$. $\{$<digit>$\}_0^{9-n}$

Note: A decimal number could not be defined as

<decimal number> := <integer>.<integer>

since (a) no more than nine digits are permitted in a number, whereas the above construct would allow a maximum of 18 and (b) since an <integer> must contain at least one digit, the form $\{$<digit>$\}$. **cannot be generated.**

<sign> := <null>|−|+
<exponent> := E<sign>$\{$<digit>$\}_1^2$
<number> := $\{$<integer>|<fraction>|<decimal number>$\}_1^1$ $\{$<exponent>$\}_0^1$
<signed number> := <sign><number>
<simple variable> := <alphabetic character>$\{$<digit>$\}_0^1$
<subscripted variable> := <alphabetic character>(<expression>$\{$,<expression>$\}_0^{10}$)
<variable> := <simple variable>|<subscripted variable>
<function name> := SIN|COS|TAN|ATN|EXP|ABS|LOG|SQR|INT|RND|FN<alphabetic character>
<function term> := <function name>(<expression>)
<term> := <number>|<variable>|<function term>|(<expression>)
<involution factor> := <term>|<involution factor>↑<term>
<multiply factor> := <involution factor>|<multiply factor>$\{$*|/$\}_1^1$<involution factor>
<expression> := <multiply factor>|<sign><expression>|<expression>$\{$+|−$\}_1^1$<multiply factor>

<assignment statement> := LET<variable>=<expression>
<read list> := <variable>$\{$,<variable>$\}_0^\infty$
<READ statement> := READ<read list>
<number list> := <signed number>$\{$,<signed number>$\}_0^\infty$
<DATA statement> := DATA<number list>
<message> := "$\{$<alphabetic character>|<digit>|<special character>$\}_1^{\infty}$"
<print item> := <expression>|<message>|<message><expression>
<print list> := <null>|<print item>$\{$,<print item>$\}_0^\infty\{$,$\}_0^1$
<PRINT statement> := PRINT<print list>
<line no 1> := $\{$<digit>$\}_1^3$
<GO TO statement> := GO$\{$b$\}_0^1$TO<line no 1>
<comment> := REM$\{$<alphabet character>|<digit>|<special character>$\}_0^\infty$
<relation op> := >=|>|<>|<|<=|=
<IF statement> := IF<expression><relation op><expression>THEN <line no 1>
<FOR statement> := FOR<simple variable>=<expression>TO<expression> $\{$STEP<expression>$\}_0^1$
<NEXT statement> := NEXT<simple variable>
<END statement> := END
<size> := <integer>$\{$,<integer>$\}_0^1$
<dimension variable> := <alphabetic character>(<size>)
<DIMension statement> := DIM<dimension variable> $\{$,<dimension variable>$\}_0^\infty$
<DEFine statement> := DEFbFN<alphabetic character>(<simple variable>) =<expression>
<GOSUB statement> := GOSUB<line no 1>
<RETURN statement> := RETURN
<statement body> := <assignment statement>|<READ statement>|<DATA statement>|<PRINT statement>|<GO TO statement>|<IF statement>|<FOR statement>|<NEXT statement>|<DIMension statement>|<DEFine statement>|<GOSUB statement>|<RETURN statement>|<comment>
<line number> := $\{$<digit>$\}_1^3$b
<BASIC statement> := <line number><statement body>
<BASIC program> := $\{$<BASIC statement>$\}_1^\infty$ <line number><END statement>

APPLICATION IV : Evaluation du langage oral d'handicapés mentaux.

1. Définition du problème.

Au cours de l'année, nous avons eu la chance de rencontrer Monsieur Emile Counet, inspecteur de français de l'enseignement secondaire spécial, et nous avons pu, grâce à sa collaboration, appliquer les tables séquentielles à un problème concret.

Monsieur Counet s'intéresse aux enfants handicapés mentaux et étudie les méthodes qu'utilisent les éducateurs pour leur apprendre à parler. Il tente d'évaluer ces méthodes c'est-à-dire déterminer si elles sont bien adaptées à ces enfants et ce en suivant leurs progrès.

Pour déterminer le niveau "oral" atteint par un enfant, plusieurs indices sont calculées à partir des documents oraux receuillis :

- indice d'endurance (nombre total de syllabes divisé par le nombre de chaînes parlées);
- indice de cohérence (indice plus qualitatif qui rend compte de la coordination et de la logique des phrases).

D'autre part, ce niveau oral dépend aussi de la complexité structurale des chaînes parlées. A cet effet Monsieur Counet a su dégager l'esquisse d'une échelle des structures du langage oral de ces enfants (voir annexe).
Grâce à cette échelle, il est possible de déterminer un indice de complexité structurale par détermination du niveau moyen des chaînes parlées.

Le problème résolu dans cette application est la déterminations de manière systématique, du niveau d'une chaîne parlée quelqu'elle soit, en accord avec l'échelle.

Afin de résoudre ce problème, nous élaborerons des tables séquentielles qui à partir des unités (en général les mots) qui composent la chaîne parlée, comme données (conditions d'entrée) décident du niveau de la chaîne.

2. Elaboration des tables séquentielles.

Par examen de l'échelle, nous observons 2 types de structure :

- les structures rigides (le groupe sujet et prédicat sont confondus);

- les structures vivantes (le niveau de la chaîne est rendu par le niveau du ou des groupes sujet et le niveau du groupe prédicat).

A. Structures rigides.

Nous créerons une table séquentielle (table T) pour la détermination du niveau des phrases rigides.

En en-tête de la table figureront les mots (nom commun, adjectif ...) et groupes de mots (complément déterminatif, forme impersonnelle ....) pouvant composer la chaîne parlée. A partir de l'échelle, il suffit d'introduire dans la table les phases nécessaires à l'enregistrement de toutes les séquences possibles correspondant aux différents niveaux et une phase par niveau (celles-ci sont généralement atteintes après décodage du symbole fin de chaîne parlée "*" et sont toutes phases finales; la décision à laquelle elles correspondent décide du niveau de la chaîne parlée d'entrée).

B. Structures vivantes.

Il convient de dissocier les groupes sujet des groupes prédicat.

Le niveau des groupes sujet sera évalué à partir d'une table séquentielle propre aux sujets (table A). A nouveau, il s'agit d'introduire en en-tête les mots et groupes de mots pouvant composer un groupe sujet ainsi que les phases nécessaires à l'enregistrement des séquences et celles correspondant aux différents niveaux définis par l'échelle.

Après examen de l'échelle, nous avons décidé d'élaborer 4 tables séquentielles (tables B, C, D, G) pour déterminer le niveau prédicat des chaînes vivantes, de manière à établir une partition sur l'ensemble des niveaux par la relation d'équivalence : "2 niveaux sont équivalents si les chaînes de ces niveaux admettent les mêmes unités de découpe" (ces unités peuvent être le mot ou groupe de mots, le complément, la proposition etc...).

La chaîne parlée considérée sera aiguillée vers l'une ou l'autre de ces tables en fonction de sa forme générale :

1. Est-elle simple (une proposition principale) ou multiple (plusieurs) ? Si elle est multiple, elle est aiguillée vers la table D, caractéristique du niveau VIII. Les unités constituantes de la chaîne sont les phrases toutes entières et les conjonctions
   Sinon

2. Comporte-t-elle ou non des propositions subordonnées ?
   Si oui, elle est aiguillée vers la table G, caractéristique du niveau VIII et ses unités de découpe sont les propositions
   Sinon

3. comporte-t-elle un ou plusieurs compléments ?
   Si elle en a plusieurs, elle est dirigée vers la table C, caractéristique des niveaux IV, V, VI et ses unités de découpe sont les compléments, verbes et conjonctions, sinon elle est aiguillée vers la table B caractéristique des niveaux, I, II, III et ses unités de découpe sont les mots et groupes de mots.

Nous pouvons résumer ce qui précède au moyen de la table combinatoire.

Structure : 1. rigide 2. vivante	1	2	2	2	2	2	2	2
Forme : 1. énonciative, négative, impérative 2. interrogative 3. passive 4. emphasée	–	1	1	1	1	2	3	4
Proposition principale : 1. simple 2. multiple	–	1	1	1	2	–	–	–
Proposition subordonnée ? 1. oui 2. non	–	2	2	1	–	–	–	–
Complément : 1. un 2. plusieurs	–	1	2	–	–	–	–	–
Découper la chaîne en unités qui sont les mots et groupes de mots suivants : forme impersonnelle-présentatif- et déterminatif Goto table T	×							
Prélever les groupes sujet. Découper les en unités qui sont les mots et groupes de mots suivants : ct apposé- ct déterminatif présentatif-gérondif antéposé Do table A		×	×	×	×	×	×	×
Découper la chaîne en unités qui sont les phrases et conjonctions Goto table D					×			
Découper la chaîne en unités qui sont les propositions Goto table G				×				
Découper la chaîne en unités qui sont les verbes, compléments, conjonctions Goto table C			×					
Découper la chaîne en unités qui sont les mots suivants : attribut du sujet – attribut du complément – complément déterminatif Goto table B		×						
Le niveau prédicat de la chaîne est Xa						×		
Le niveau prédicat de la chaîne est Xc							×	
Le niveau prédicat de la chaîne est Xd								×

Remarque :

Si la chaîne est passive, interrogative ou emphasée, le niveau prédicat est connu immédiatement.

Les tables séquentielles T, A, B, C, D, G sont les suivantes :

Table T — Détermination du niveau des phrases rigides.

	Phases	Niveau	Symbole fin de chaîne parlée	Présentatif	Nom commun	Nom propre	Article défini	Article indéfini	Article partitif	Complément déterminatif	Forme impersonnelle	Possessif	Démonstratif	Adjectif
			*	PRES	NC	NP	AD	AI	AP	CD	FI	P	D	ADJ
	1		50	9	2	2	3	4	5	50	17	50	50	50
Mot phrase	2		41	50	50	50	50	50	50	50	50	50	50	50
Article défini	3				6									
Article indéfini	4				7									
Article partitif	5				8									
Article défini + nom comm.	6		42											
Article indéfini + nom comm.	7		43											
Article partitif + nom comm.	8		44											
Article présentatif	9				10	11	11	11				15	15	
Présentatif + nom comm.	10		45											
Présentatif + article	11				12									13
Présentatif + article + nom commun	12		46						14					
Présentatif + article + adjectif	13				14									
Présentatif + article + adj. + nom comm. / Présentatif + art. + nom comm. +ct. dét.	14		47											
Présentatif + possessif ou démonstratif	15				16									
Présentatif + poss. ou démonst. +nom comm.	16		48											
Forme impersonnelle	17		49											
Mot phrase *	41	-Ia												
Article défini + nom comm. *	42	-Ic1												
Article indéfini + nom comm. *	43	-Ic2												
Article partitif + nom comm. *	44	-Ic3												
Présentatif + nom comm. *	45	-Id1												
Présentatif + art. + nom comm. *	46	-Id2												
Présentatif + art. + nom comm. + ct dét. * / Présentatif + art. + adj. + nom comm. *	47	-Id4												
Présentatif + possessif + nom comm. * / démonstratif	48	-Id3												
Forme impersonnelle *	49	-Ie												
	50	non défini												

Table A — Détermination du niveau des groupes sujet.

	PHASES	*	CONJ	NPL	NP	PRELAS	NC	PP	DET	PREART	POSTART	ADJ	CD	CA	PRQUI	PRQUE	PR	GA	PA	PRES
	1	100	136	2	5	136	136	7	8	26	136	136	136	136	136	136	136	135	131	3
Nom propre locuteur	2	101	133												127	128	129			
Présentatif	3				4				10	30										
Prés. +nom propre ≠ locuteur	4	102	133												130					
Nom propre ≠ locuteur	5	104	133			6									127	128	129			57
Nom prop ≠ loc. + pron. relais	6	103																		
Pronom	7	105	133																	
Déterminant	8						9				24	13								
Dét. + nom comm.	9	106	134			12						18	21	31	127	128	129			57
Présentatif + dét.	10						11				28	15								
Prés + dét + nom comm.	11	107	134										22	32	130					
Dét + nom comm.+ pron. relais	12	108																		
Dét + adj	13						14													
Dét + adj + nom comm.	14	109	134			17							45	46	127	128	129			
Prés + dét + adj	15						16													
Prés + dét + adj + nom comm.	16	110	134										46	46	130					
Dét + adj + nom comm. + p. rel.	17	111																		
Dét + nom comm. + adj.	18	109	134										47	47	127	128	129			
Prés + dét + nom comm.+ adj	19	113	134			20							48	48	130					
Dét + nom comm. + adj. + p. rel.	20	111																		
Dét + nom comm. + ct dét.	21	115	134			23								49	127	128	129			
Prés + dét + nom comm. + ct dét.	22	116	134											50	130					
Dét + nom comm. + ct dét + p. rel.	23	117																		
[Préart] + dét + [postart]	24						25					34								
[Préart] + dét + [postart] + nom comm.	25	118	134			27						39	40	42	127	128	129			
Préarticle	26								24											
[Préart] + dét + [postart] + nom comm. + p. relais	27	119																		
Prés + [préart] + dét + [postart]	28						29					37								
Prés + [préart] + dét + [postart] + nom comm.	29	120	134									38	41	43	130					
Prés + préart	30								28											
Dét. + nom comm. + ct app.	31	121	134			33									127	128	129			
Prés + dét + nom comm. + ct app.	32	122	134												130					
Dét + nom comm. + ct app. + p. rel.	33	123																		
[Préart] + dét + [postart] + adj	34						35													
[Préart] + dét + [postart] + adj. + nom comm.	35	124	134			36							51	51	127	128	129			
[Préart] + dét + [postart] + adj + nom comm. + p. relais	36	125																		
Prés + [préart] + dét + [postart] +adj.	37						38													
Prés + [préart] + dét + [postart] + adj + nom comm.	38	126	134										52	52	130					
[Préart] + dét + [postart] + nom comm. + adj	39	124	134			36							55	55	127	128	129			
Prés + [préart] + dét + [postart]+ nom comm. + adj	40	126	134										56	56	130					
[Préart] + dét + [postart] + nom comm. + ct dét.	41	124	134			36								53	127	128	129			
Prés + [Préart] + dét + [postart] + nom comm. + ct dét.	42	126	134											54	130					
[Préart] + dét + [postart] + nom comm. + ct app.	43	124	134			36									127	128	129			
Prés + [Préart] + dét + [postart] + nom comm. + ct app.	44	126	134																	
Dét + adj + {ct app / ct dét}	45	124	134			36														
Prés + dét + adj + {ct app / ct dét}	46	126	134																	
Dét + nom comm. + adj + {ct app / ct dét}	47	124	134			36														
Prés + dét + nom comm. + adj + {ct app / ct dét}	48	126	134																	
Dét + nom comm. + ct dét + ct app	49	124	134			36														
Prés + dét + nom comm. + ct dét + ct app	50	126	134																	
[Préart]+dét + [postart] + adj + nom comm + {ct dét / ct app}	51	124	134			36														
Prés + [préart] + dét + [postart] + adj + nom comm. + {ct app / ct dét}	52	126	134																	
[Préart] + dét + [postart] + nom comm. + ct dét + ct app.	53	124	134			36														
Prés + [préart] + dét + [postart] + nom comm. +ct dét+ctapp	54	126	134																	
[Préart] + dét + [postart] + nom comm. + adj + {ct app / ct dét}	55	124	134			36														
Prés + [préart] + dét + [postart] + nom comm. +adj+ {ct app / ct dét}	56	126	134																	
Dét + nom comm. + présentatif nom propre + présentatif	57	132																		

Les phases finales qui déterminent le niveau des groupes sujet sont :

	PHASES	NIVEAU
Pas de sujet (*)	100	0
Nom propre locuteur *	101	I
Prés + nom propre ≠ locuteur *	102	IIa1
Nom propre ≠ loc + pr rel. *	103	IIa2
Nom propre ≠ locuteur *	104	IIa3
Pronom *	105	IIb
Dét + nom comm. *	106	III 3
Prés + dét + nom comm. *	107	III 1
Dét + nom comm. + p. rel. *	108	III 2
Dét + adj + nom comm. *	109	IV a3
Prés + dét + adj + nom comm. *	110	IV a1
Dét + adj + nom comm. + p. rel. *	111	IV a2
Dét + nom comm. + adj. *	113	IV a4
Dét + nom comm. + ct dét. *	115	IV b3
Prés + dét + nom comm. + ct dét. *	116	IV b1
Dét. + nom comm. + ct dét + p. rel. *	117	IV b2
[Préart] + dét. + [postart] + nom comm*	118	IV c3
[Préart] + dét + [postart] + nom comm. + p. rel. *	119	IV c2
Prés + [préart] + dét + [postart] + nom comm. *	120	IV c1
Dét + nom comm. + ct app. *	121	IV d3
Prés + dét + nom comm. + ct app. *	122	IV d1
Dét + nom comm. + ct app. + p. rel. *	123	IV d2
[Préart] + dét + [postart] + adj + nom comm. *	124	IV e3
[Préart] + dét + [postart] + adj + nom comm. + p. rel. *	125	IV e2
Prés + [préart] + dét + [postart] + adj + nom comm. *	126	IV e1
------ pronom rel. qui -----	127	V a
------ pronom rel. que -----	128	V b
------ pronom rel. ≠ qui, que ----	129	V c
Présentatif--- p. rel. qui ---	130	VI c
Participe antéposé ---	131	VI b

	PHASES	NIVEAU
Dét + nom comm. + présentatif * / Nom propre + présentatif *	132	VI d
Sujet simple + conjonction ---	133	IV g
Sujet expansé + conjonction ---	134	IV h
Gérondif antéposé ---	135	VI a
	136	cas non défini

Table B — Détermination du niveau des groupes prédicat lorsque la chaîne est simple et composée d'un seul complément.

	Phases	*	AUX	PP	V	PRON	DET	NC	NP	PREP	INF	ADJ	AS	AC	CD	ADV	ST	NST
	1		31		2	3				14						36		
Verbe (ou aux. + part. passé)	2	51				19	5		7	9	2		8			33		
Pronom	3		34		4													
Pronom + verbe	4	52									29							
Verbe + dét	5							6				25						
Verbe + dét + nom comm.	6	53									30	27			28			
Verbe + nom propre	7	53									30							
Verbe + attr. du sujet Verbe + prép + attr. du sujet	8	53																
Verbe + prép.	9					19	10		11		13		8					
Verbe + prép + dét	10							12				25						
Verbe + prép + nom propre	11	54												23				
Verbe + prép + dét + nom comm.	12	54								24		27		23	28			
Verbe + prép + infinitif	13	55					21		20	22								
Prép.	14						15		16									
Prép. + dét	15							16										
Prép + { dét + nom comm. nom propre	16																17	63
Idem stéréotypé	17		35		18													
Idem + verbe	18	56																
Verbe + [prép] + pronom	19										20							
Verbe + [prép] + pron + inf.	20	57																
Verbe + prép + inf + [prép] + dét	21							20										
Verbe + prép + inf + prép	22						21		20									
Verbe + prép + { nom propre + [prép] + attr du ct. dét+nom comm.	23	58																
Verbe + prép + { nom propre + prép. dét + nom comm.	24													23				
Verbe + [prép] + dét + adj	25							26				25						
Verbe + [prép] + dét + adj + nom comm.	26	59										27			28			
Verbe + [prép] + dét + nom comm. + adj	27	59													28			
Verbe + [prép] + dét + nom comm. + ct dét	28	60																
Pronom + verbe + inf	29	61																
Verbe + { nom propre Dét + nom comm. + inf	30	62																
Auxiliaire	31			2												32		
Auxiliaire + adverbe	32			33														
{ Verbe Aux + part. passé + adv	33	56																
Pronom + aux.	34			4														
Prép + { dét + nom comm. nom propre + aux	35			18														
Adverbe	36		37		33													
Adverbe + aux	37			33														

Les phases finales qui déterminent le niveau des groupes prédicat sont :

	PHASES	NIVEAU
{ Verbe { Aux + participe passé *	51	I
Pronom + verbe *	52	II d
Verbe + { nom propre { dét + nom comm. *	53	II a
ou		
Verbe + [prép] + attr sujet *		
Verbe + prép + { nom propre { dét + nom comm. *	54	II b
Verbe + prép + inf *	55	II c
Prép + { nom propre { dét + nom comm. + verbe *	56	II e
Verbe + [prép] + pron + inf *	57	III c
Verbe + prép + { nom propre { dét + nom comm. + [prép] + attr ct *	58	III d
Verbe + [prép] + dét + { adj + nom comm { nom comm. + adj *	59	III a
Verbe + [prép] + dét + nom comm. + ct dét. *	60	III b
Prnom + verbe + inf *	61	VIII e
Verbe + { nom propre { dét + nom comm. + inf *	62	VIII d
Prép + { nom propre { dét + nom comm. non stéréotypé ...	63	IX
	64	cas non défini

Table C — Détermination du niveau du groupe prédicat lorsque la chaîne est simple et composée de plusieurs compléments.

	Phases	fin de chaîne *	verbe V	complément simple CS	complément expansé CE	conjonction CONJ
	1	58	2	12	15	58
Verbe	2			3	7	
Verbe + ct simple	3			4	9	5
Verbe+ct simple+ct simple	4	51		4		5
Verbe+ct(s)simple(s)+conj.	5			6	9	
Verbe+ct(s)simple(s)+conj. + ct simple	6	52				5
Verbe + ct expansé	7			8	11	10
Verbe+ct expansé+ct simple	8	53				
Verbe+ct simple+ct expansé	9	54				
Verbe+ct expansé + conj.	10			8	11	
Verbe+ct(s) expansé(s)	11	55				
Ct simple	12		16	13		18
Ct(s) simple(s)	13		14			
Ct(s) simple(s) + verbe	14	56				
Ct expansé	15		16			18
Ct expansé + verbe	16			17	17	
{Ct simple / Ct simple} + {ct simple / ct expansé}	17	57				
{Ct simple / Ct expansé} + conj.	18			19	19	
{Ct simple / Ct expansé} + conj. + {ct simple / ct expansé}	19		17			

Les phases finales qui déterminent le niveau des groupes prédicats sont :

	PHASES	NIVEAU
Verbe + ct(s) simple(s) *	51	IVa
Verbe+ct(s) simple(s) + conj + ct(s) simple(s) *	52	IVb
Verbe+ct expansé + ct simple *	53	Vb
Verbe+ ct simple + ct expansé *	54	Va
Verbe+ct(s) expansé (s) *	55	VIe
Ct(s) simple(s) + verbe	56	IVc
{Ct simple / Ct expansé} + verbe + {ct simple / ct expansé} *	57	IXe
	58	cas non défini

Table D — Détermination du niveau du groupe prédicat lorsque la chaîne est multiple.

		fin de chaîne	phrase	conjonction
	PHASES	*	P	CONJ
	1	13	2	13
Phrase	2	13	3	4
Phrase + phrase	3	11	3	4
Phrase + conj.	4	13	5	13
Phrase + conj. + phrase	5	12		4

Les phases finales sont :

	PHASES	NIVEAU
phrase + phrase *	11	VII a
phrase + conjonction + phrase *	12	VII b
	13	cas non défini

Table G

Détermination du niveau du groupe prédicat lorsque la chaîne est simple et comporte au moins une proposition subordonnée.

		fin de chaîne	proposition principale	proposition complément circonstant.	proposition relative qui	proposition relative que	proposition conjonctive intr.	proposition interrogative indir.
	Phases	*	PP	PCC	PRQUI	PRQUE	PCI	PII
	1	16	2	8	16	16	16	16
Prop. princ.	2			3	4	5	6	7
Prop. cc + prop. princ. ou								
Prop princ + prop cc	3	11						
Prop princ + prop rel.qui	4	12						
Prop princ + prop rel.que	5	13						
Prop princ + prop conj intr.	6	14						
Prop princ + prop int.ind	7	15						
Prop cc	8		3					

Les phases finales sont :

	PHASES	NIVEAU
Prop princ + prop cc *	11	VIIIg
Prop princ + prop rel. qui *	12	VIII a
Prop princ + prop rel. que *	13	VIII b
Prop princ + prop conj intr *	14	VIII c
Prop princ + prop int. ind. *	15	VIII f
	16	cas non défini

Remarques :

1. lorsque pour une phase présente et une condition d'entrée la phase suivante n'est pas spécifiée explicitement, il s'agit de celle qui est associée aux cas non définis par l'échelle;

2. en entrée, les conditions sont les unités qui composent la chaîne en séquence, en sortie, la table décide du niveau soit du groupe sujet, soit du groupe prédicat, soit de la chaîne, selon le cas;

3. dans la table A, nous observons des phases équivalentes en effet :

$$43 \equiv 45 \equiv 47 \equiv 49 \equiv 51 \equiv 53 \equiv 55$$
$$44 \equiv 46 \equiv 48 \equiv 50 \equiv 52 \equiv 54 \equiv 56$$

Ces phases peuvent donc être fusionnées.

3. Implémentation du problème.

A. Programme.

Nous voulons à présent implémenter cette application c'est-à-dire rendre le processus de consultation des tables automatique Nous désirons, de plus, rendre ce processus conversationnel à partir d'un terminal.

Pour chaque chaîne orale, l'ordinateur (le programme que nous allons rédiger) pose une série de questions à l'utilisateur qui est au terminal afin de connaître la forme générale de la chaîne et de déterminer la ou les tables à consulter. A partir de là, l'ordinateur spécifie à l'utilisateur les unités selon lesquelles il doit découper la chaîne, le groupe sujet ou le groupe prédicat et lui demande de les introduire au terminal. Ainsi le programme, après consultation de la table adéquate, avec les unités d'entrée reçues, peut déterminer le niveau demandé.

Nous voyons donc que le programme à rédiger comporte 3 parties :

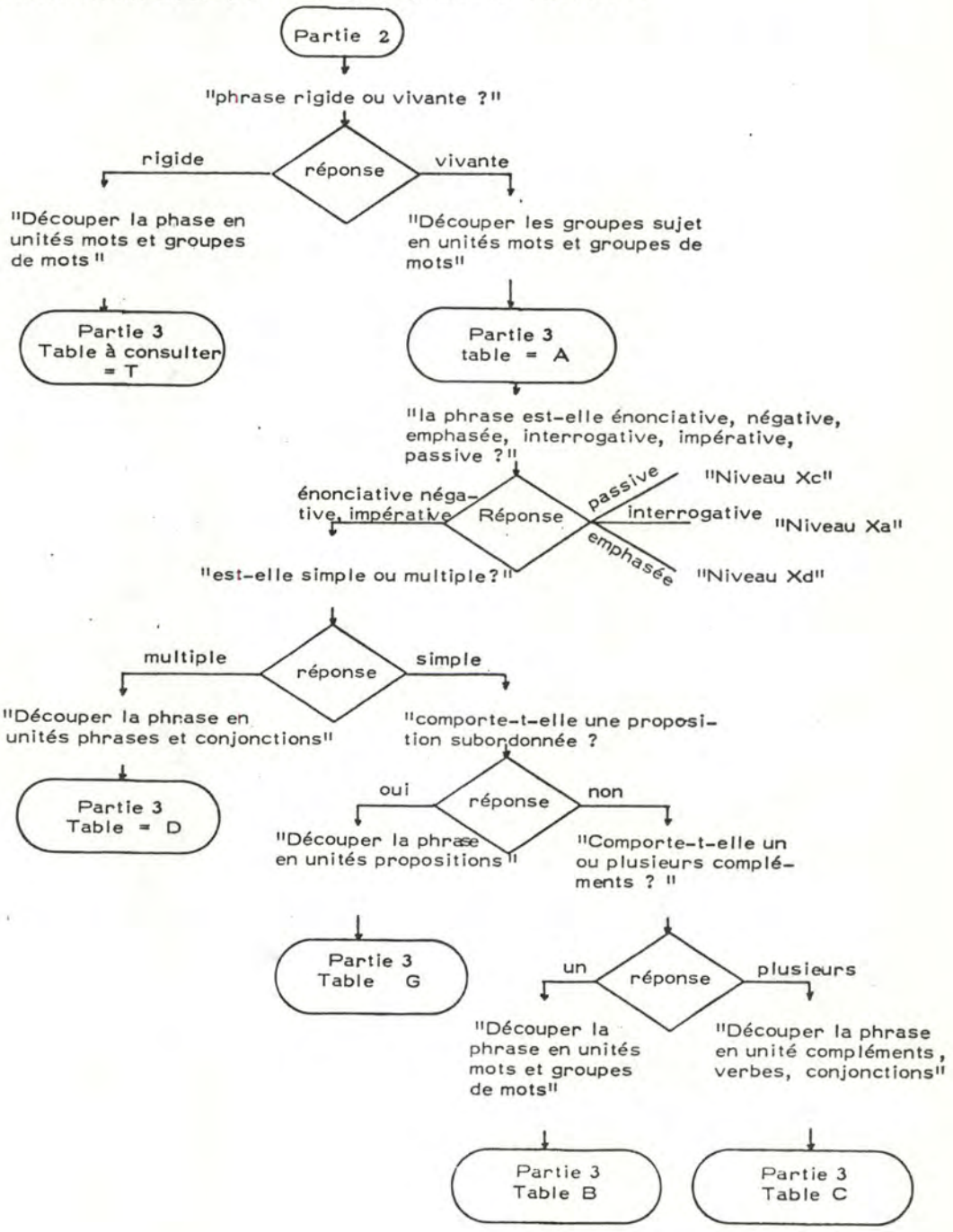
1. le chargement des tables;
2. le dialogue avec l'utilisateur au terminal;
3. la consultation des tables.

Partie 1 :

Elle n'offre aucune difficulté; il suffit de lire, d'introduire les tables définies par le problème en mémoire.

Partie 2 :

Il faut programmer les différentes questions qu'il faut poser à l'utilisateur afin de déterminer la ou les tables à consulter. L'enchaînement de ces questions dépend des réponses fournies par l'utilisateur et est schématisé par l'organigramme suivant :



Partie 3 :

Toute table définie par le problème est implémentée sous forme d'un tableau. L'indice ligne E du tableau correspond à la phase E de la table; ceci est possible par le fait que les n° de phase se succèdent comme les nombres entiers à partir de 1. Si cela n'avait pas été le cas, il aurait suffi de considérer le vecteur des phases et d'établir une bijection entre celui-ci et les indices du tableau ou encore d'introduire une ligne vide dans le tableau pour tout indice E tel que E ne soit pas phase.

Toutes les tables sont consultées selon le même algorithme :

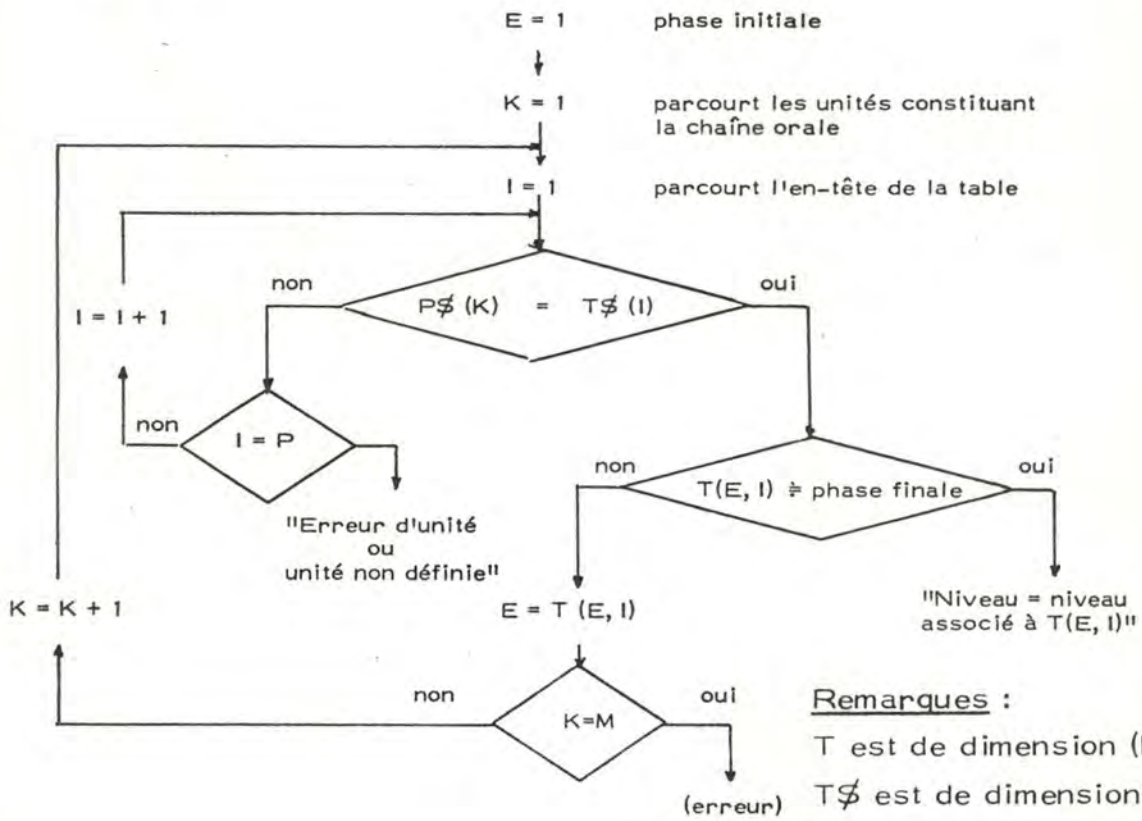
- soit la table séquentielle T : ses éléments sont référencés par

T (E , I).

indice ligne indice colonne

- soit le vecteur T$ de l'en-tête de cette table
- soit le vecteur P$ des unités introduites par l'utilisateur et constituant le groupe sujet, le groupe prédicat ou la chaîne à évaluer; soit K l'indice de P$.

L'algorithme de consultation et de détermination des niveaux est le suivant :



Remarques :

T est de dimension (N, P)

T$ est de dimension (P)

P$ est de dimension (M)

A . Problème des chaînes de prolongation.

Un petit problème doit être examiné de plus près : lorsqu'un enfant handicapé parle, il n'a certes pas un débit aussi rapique le nôtre et bien souvent il marque des temps d'arrêt avant d'avoir exprimé une phrase complète (au sens de la grammaire classique); de là le problème de savoir si la chaîne parlée à analyser n'est pas une prolongation de la chaîne orale précédente.

Exemple : j'ai vu ..... (temps d'arrêt) (1)

un film . (2)

Pour cette raison, la première question que posera l'ordinateur à l'utilisateur sera : la phrase suivante est-elle une prolongation de la phrase précédente.

Si tel est le cas, et que cette nouvelle chaîne revêt la forme d'une proposition (principale ou subordonnée) ou d'un complément supplémentaire, il faudra que l'utilisateur regroupe la phrase précédente et la phrase présente en une pour en déterminer le niveau global (ceci pour une raison d'unités de découpe). Par contre si la phrase de prolongation revêt une autre forme, il suffit de la découper selon les mêmes unités que la phrase précédente et d'introduire ces unités; on obtiendra ainsi aussi le niveau global. Dans ce cas, le programme de consultation continue à procéder sur la table déterminée par la phrase précédente, à partir de la phase qui était phase présente avant le décodage du symbole fin de chaîne orale "*" .

Dans le cas de l'exemple, nous obtiendrions le niveau de "jai vu" (niveau sujet + niveau prédicat) et le niveau de "j'ai vu un film" (même niveau sujet + niveau prédicat plus élevé) qui exprime l'effort fait par l'enfant.

B . Cas non définis par l'échelle.

Lorsqu'une chaîne orale ne correspond à aucun niveau défini par l'échelle, le programme le signalera; il donnera en plus la table consultée, la phase et la condition qui ont provoqué le branchement à la phase finale "cas non défini". Ceci permettra de remanipuler ultérieurement l'échelle et de voir s'il ne serait pas nécessaire de définir de nouveaux niveaux.

```
 1 DIM P$(15),T(17,12),T$(12),A(44,19),A$(19),B(38,17),B$(17)
 2 DIM C(19,5),C$(5),D(5,3),D$(3),G(8,7),G$(7)
10 FOR I=1 TO 17
11 FOR J=1 TO 12
12 READ T(I,J)
13 NEXT J
14 NEXT I
15 FOR I=1 TO 12
16 READ T$(I)
17 NEXT I
18 FOR I=1 TO 44
19 FOR J=1 TO 19
20 READ A(I,J)
21 NEXT J
22 NEXT I
23 FOR I=1 TO 19
24 READ A$(I)
25 NEXT I
26 FOR I=1 TO 37
27 FOR J=1 TO 17
28 READ B(I,J)
29 NEXT J
30 NEXT I
31 FOR I=1 TO 17
32 READ B$(I)
33 NEXT I
34 FOR I=1 TO 19
35 FOR J=1 TO 5
36 READ C(I,J)
37 NEXT J
38 NEXT I
39 FOR I=1 TO 5
40 READ C$(I)
41 NEXT I
42 FOR I=1 TO 5
43 FOR J=1 TO 3
44 READ D(I,J)
45 NEXT J
46 NEXT I
47 FOR I=1 TO 3
48 READ D$(I)
49 NEXT I
50 FOR I=1 TO 8
51 FOR J=1 TO 7
52 READ G(I,J)
53 NEXT J
54 NEXT I
55 FOR I=1 TO 7
56 READ G$(I)
57 NEXT I
60 PRINT
61 PRINT "DETERMINATION DU NIVEAU DE LA PHRASE SUIVANTE"
62 PRINT "LA PHRASE SUIVANTE EST-ELLE UNE TENTATIVE DE PROLONGATION "
63 PRINT "DE LA PHRASE PRECEDENTE?OUI-NON"
64 INPUT R$
66 IF R$="NON" THEN 280
67 PRINT "REVET-ELLE LA FORME D-UN COMPLEMENT,D'UNE PROPOSITION"
68 PRINT "SUBORDONNEE OU D'UNE PROPOSITION PRINCIPALE? OUI-NON"
70 INPUT R$
80 IF R$="NON" THEN 180
90 PRINT "REGROUPER LA PHRASE ORALE PRECEDENTE ET LA PHRASE"
91 PRINT " PRESENTE EN UNE"
```

```
100 PRINT "Y-A-T-IL DE NOUVEAUX GROUPES SUJET ? OUI-NON"
110 INPUT R$
120 IF R$="NON" THEN 1430
150 PRINT "PRELEVER LE 1ER GROUPE SUJET A ANALYSER"
160 J=J+1
170 GOTO 510
180 PRINT "DECOUPER LA PHRASE SELON LES MEMES UNITES QUE LA"
181 PRINT " PHRASE PRECEDENTE"
190 PRINT "INTRODUISEZ LE NOMBRE D'UNITES"
200 INPUT N
210 PRINT "INTRODUISEZ LES UNITES EN SEQUENCE, CLOTUREE PAR *"
220 FOR I=1 TO N
230 INPUT P$(I)
240 NEXT I
250 E=M
260 K=1
270 GOTO 2590
280 PRINT "LA PHRASE EST-ELLE RIGIDE-VIVANTE?"
290 INPUT R$
300 IF R$="VIVANTE" THEN 490
310 PRINT "DECOUPER LA PHRASE EN UNITES QUI SONT LES MOTS"
311 PRINT " ET GROUPES DE MOTS SUIVANTS : PRESENTATIF - "
312 PRINT "FORME IMPERSONNELLE - COMPLEMENT DETERMINATIF"
320 PRINT "INTRODUISEZ LE NOMBRE D'UNITES"
330 INPUT N
340 PRINT "INTRODUISEZ LES UNITES EN SEQUENCE CLOTUREES PAR *"
350 FOR I=1 TO N
352 INPUT P$(I)
354 NEXT I
356 K=E=1
358 FOR I=1 TO 12
360 IF T$(I)=P$(K) THEN 364
362 NEXT I
364 IF T(E,I)>=41 THEN 372
366 E=T(E,I)
368 K=K+1
370 GOTO 358
372 ON T(E,I)-40 GOTO 374,378,381,383,385,387,389,391,393,395
374 PRINT "LE NIVEAU EST -1A"
376 GOTO 396
378 PRINT "LE NIVEAU EST -1C1"
380 GOTO 396
381 PRINT "LE NIVEAU EST -1C2"
382 GOTO 396
383 PRINT "LE NIVEAU EST -1C3"
384 GOTO 396
385 PRINT "LE NIVEAU EST -1D1"
386 GOTO 396
387 PRINT "LE NIVEAU EST -1D2"
388 GOTO 396
389 PRINT " LE NIVEAU EST -1D4"
390 GOTO 396
391 PRINT "LE NIVEAU EST -1D3"
392 GOTO 396
393 PRINT "LE NIVEAU EST -1E"
394 GOTO 396
395 PRINT "LE CAS N-EST PAS DEFINI:TABLE = T,E = ";E;"I = ";I
396 GOTO 60
490 J=1
500 PRINT "PRELEVER LE 1ER GROUPE SUJET DE LA PHRASE"
510 PRINT "DECOUPEZ-LE EN UNITES QUI SONT LES MOTS ET GROUPE "
511 PRINT "DE MOTS SUIVANTS : COMPLEMENT DETERMINATIF - COMPLEMENT "
512 PRINT "APPOSE - PRESENTATIF - GERONDIF ANTEPOSE"
520 PRINT "INTRODUISEZ LE NOMBRE D'UNITES"
530 INPUT N
```

```
540 PRINT "INTRODUISEZ LES UNITES EN SEQUENCE CLOTUREES PAR *"
550 FOR I=1 TO N
560 INPUT P$(I)
570 NEXT I
580 K=E=1
590 FOR I=1 TO 19
600 IF A$(I)=P$(K) THEN 620
610 NEXT I
620 IF A(E,I)>77 THEN 660
630 E=A(E,I)
640 K=K+1
650 GOTO 590
660 PRINT "NIVEAU DU GROUPE SUJET ";J;" = ";
665 IF A(E,I)>110 THEN 675
670 ON A(E,I)-100 GOTO 710,730,750,770,790,810,830,850,870,890
675 IF A(E,I)>120 THEN 685
680 ON A(E,I)-110 GOTO 910,910,930,930,950,970,990,1010,1030,1050
685 IF A(E,I)>130 THEN 700
690 ON A(E,I)-120 GOTO 1070,1090,1110,1130,1150,1170,1190,1210,1230,1250
700 ON A(E,I)-130 GOTO 1270,1290,1310,1330,1350,1370
710 PRINT "1"
720 GOTO 1380
730 PRINT "2A1"
740 GOTO 1380
750 PRINT "2A2"
760 GOTO 1380
770 PRINT "2A3"
780 GOTO 1380
790 PRINT "2B"
800 GOTO 1380
810 PRINT "33"
820 GOTO 1380
830 PRINT "31"
840 GOTO 1380
850 PRINT "32"
860 GOTO 1380
870 PRINT "4A3"
880 GOTO 1380
890 PRINT "4A1"
900 GOTO 1380
910 PRINT "4A2"
920 GOTO 1380
930 PRINT "4A4"
940 GOTO 1380
950 PRINT "4B3"
960 GOTO 1380
970 PRINT "4B1"
980 GOTO 1380
990 PRINT "4B2"
1000 GOTO 1380
1010 PRINT "4C3"
1020 GOTO 1380
1030 PRINT "4C2"
1040 GOTO 1380
1050 PRINT "4C1"
1060 GOTO 1380
1070 PRINT "4D3"
1080 GOTO 1380
1090 PRINT "4D1"
1100 GOTO 1380
1110 PRINT "4D2"
1120 GOTO 1380
1130 PRINT "4E3"
1140 GOTO 1380
1150 PRINT "4E2"
```

```
1160 GOTO 1380
1170 PRINT "4E1"
1180 GOTO 1380
1190 PRINT "5A"
1200 GOTO 1380
1210 PRINT "5B"
1220 GOTO 1380
1230 PRINT "5C"
1240 GOTO 1380
1250 PRINT "6C"
1260 GOTO 1380
1270 PRINT "6B"
1280 GOTO 1380
1290 PRINT "6D"
1300 GOTO 1380
1310 PRINT "4G"
1320 GOTO 1380
1330 PRINT "4H"
1340 GOTO 1380
1350 PRINT "6A"
1360 GOTO 1380
1370 PRINT "CAS INDETERMINE DANS LA GRAMMAIRE, TABLE= A, E= ";E;"I= ";I
1380 PRINT
1385 PRINT "Y-A-T-IL ENCORE UN GROUPE SUJET A ANALYSER?OUI-NON"
1390 INPUT R$
1400 IF R$="NON" THEN 1430
1410 J=J+1
1420 GOTO 510
1430 PRINT "LA PHRASE EST-ELLE : ENONCIATIVE - NEGATIVE - IMPERATIVE"
1431 PRINT " - PASSIVE - EMPHASEE - INTERROGATIVE ?"
1440 INPUT R$
1450 IF R$="ENONCIATIVE" THEN 1570
1460 IF R$="NEGATIVE" THEN 1560
1470 IF R$="IMPERATIVE" THEN 1570
1480 IF R$="PASSIVE" THEN 1520
1490 IF R$="EMPHASEE" THEN 1540
1500 PRINT "NIVEAU DU GROUPE PREDICAT = 10A"
1510 GOTO 60
1520 PRINT "NIVEAU DU GROUPE PREDICAT = 10C"
1530 GOTO 60
1540 PRINT "NIVEAU DU GROUPE PREDICAT =10D"
1550 GOTO 60
1560 PRINT "RAMENER LA PHRASE A-SA FORME ENONCIATIVE"
1570 PRINT "LA PHRASE EST-ELLE : SIMPLE(UNE PROPOSITION PRINCIPALE)"
1571 PRINT " OU MULTIPLE(PLUSIEURS) ?"
1580 INPUT R$
1590 IF R$="MULTIPLE" THEN 1920
1600 PRINT "LA PHRASE COMPORTE-ELLE DES PROPOSITIONS"
1601 PRINT " SUBORDONNEES ? OUI-NON"
1610 INPUT R$
1620 IF R$="NON" THEN 2150
1630 PRINT "DECOUPEZ LA PHRASE EN UNITES QUI SONT LES PROPOSITIONS"
1640 PRINT "INTRODUISEZ LE NOMBRE D'UNITES"
1650 INPUT N
1660 PRINT "INTRODUISEZ LES UNITES EN SEQUENCE CLOTUREES PAR *"
1670 FOR I=1 TO N
1680 INPUT P$(I)
1690 NEXT I
1700 K=E=1
1710 FOR I=1 TO 7
1720 IF G$(I)=P$(K) THEN 1740
1730 NEXT I
```

```
1740 IF G(E,I)>10 THEN 1780
1750 E=G(E,I)
1760 K=K+1
1770 GOTO 1710
1780 PRINT "NIVEAU DU GROUPE PREDICAT = ";
1790 ON G(E,I)-10 GOTO 1800,1820,1840,1860,1880,1900
1800 PRINT "8G"
1810 GOTO 1910
1820 PRINT "8A"
1830 GOTO 1910
1840 PRINT "8B"
1850 GOTO 1910
1860 PRINT "8C"
1870 GOTO 1910
1880 PRINT "8F"
1890 GOTO 1910
1900 PRINT "CAS INDETERMINE DANS LA GRAMMAIRE , TABLE = G , E= ";E;" I= ";I
1910 GOTO 60
1920 PRINT "DECOUPEZ LA PHRASE EN UNITES QUI SONT LES PHRASES ET CONJONCTIO
1930 PRINT "INTRODUISEZ LE NOMBRE D'UNITES"
1940 INPUT N
1950 PRINT "INTRODUISEZ LES UNITES EN SEQUENCE CLOTUREES PAR *"
1960 FOR I=1 TO N
1970 INPUT P$(I)
1980 NEXT I
1990 K=E=1
2000 FOR I=1 TO 3
2010 IF D$(I)=P$(K) THEN 2030
2020 NEXT I
2030 IF D(E,I)>10 THEN 2070
2040 E=D(E,I)
2050 K=K+1
2060 GOTO 2000
2070 PRINT "NIVEAU DU GROUPE PREDICAT = ";
2080 ON D(E,I)-10 GOTO 2090,2110,2130
2090 PRINT "7A"
2100 GOTO 2140
2110 PRINT "7B"
2120 GOTO 2140
2130 PRINT "CAS INDETERMINE DE LA GRAMMAIRE , TABLE = D , E= ";E;" I= ";I
2140 GOTO 60
2150 PRINT "LA PHRASE COMPORTE-ELLE PLUSIEURS COMPLEMENTS ? OUI-NON"
2160 INPUT R$
2170 IF R$="NON" THEN 2510
2180 PRINT "DECOUPEZ LA PHRASE EN UNITES QUI SONT LES COMPLEMENTS - "
2181 PRINT "CONJONCTION - VERBE"
2190 PRINT "INTRODUISEZ LE NOMBRE D'UNITES"
2200 INPUT N
2210 PRINT "INTRODUISEZ LES UNITES EN SEQUENCE CLOTUREES PAR *"
2220 FOR I=1 TO N
2230 INPUT P$(I)
2240 NEXT I
2250 K=E=1
2260 FOR I=1 TO 5
2270 IF C$(I)=P$(K) THEN 2290
2280 NEXT I
2290 IF C(E,I)>50 THEN 2330
2300 E=C(E,I)
2310 K=K+1
2320 GOTO 2260
2330 PRINT "NIVEAU DU GROUPE PREDICAT = ";
2340 ON C(E,I)-50 GOTO 2350,2370,2390,2410,2430,2450,2470,2490
2350 PRINT "4A"
```

```
2360 GOTO 2500
2370 PRINT "4B"
2380 GOTO 2500
2390 PRINT "5B"
2400 GOTO 2500
2410 PRINT "5A"
2420 GOTO 2500
2430 PRINT "6"
2440 GOTO 2500
2450 PRINT "4C"
2460 GOTO 2500
2470 PRINT "9"
2480 GOTO 2500
2490 PRINT "CAS INDETERMINE PAR LA GRAMMAIRE , TABLE = C , E= ";E;" I= ";I
2500 GOTO 60
2510 PRINT "DECOUPEZ LA PHRASE-EN UNITES QUI SONT LES MOTS ETGROUPES"
2511 PRINT " DE MOTS SUIVANTS : COMPLEMENT DETERMINATIF - ATTRIBUT DU"
2512 PRINT " SUJET - AUXILIAIRE DU VERBE -ADVERBE"
2520 PRINT "INTRODUISEZ LE NOMBRE D'UNITES"
2530 INPUT N
2540 PRINT "INTRODUISEZ LES UNITES EN SEQUENCE - S'IL SE PRESENTE UN"
2541 PRINT " COMPLEMENT EN TETE, FAITES SUIVRE SES UNITES PAR LES "
2542 PRINT "CARACTERES ST OU NST SELON QU'IL EST STEREOTYPE OU NON"
2543 PRINT-" - CLOTUREZ LA SEQUENCE PAR *"
2550 FOR I=1 TO N
2560 INPUT P$(I)
2570 NEXT I
2580 K=E=1
2590 FOR I=1 TO 17
2600 IF B$(I)=P$(K) THEN 2620
2610 NEXT I
2620 IF B(E,I)>50 THEN 2660
2630 E=B(E,I)
2635 M=E
2640 K=K+1
2650 GOTO 2590
2660 PRINT "NIVEAU DU GROUPE PREDICAT = ";
2665 IF B(E,I)>60 THEN 2680
2670 ON B(E,I)-50 GOTO 2690,2710,2730,2750,2770,2790,2810,2830,2850,2870
2680 ON B(E,I)-60 GOTO 2890,2910,2925,2930
2690 PRINT "1"
2700 GOTO 60
2710 PRINT "2D"
2720 GOTO 60
2730 PRINT "2A"
2740 GOTO 60
2750 PRINT "2B"
2760 GOTO 60
2770 PRINT "2C"
2780 GOTO 60
2790 PRINT "2E"
2800 GOTO 60
2810 PRINT "3C"
2820 GOTO 60
2830 PRINT "3D"
2840 GOTO 60
2850 PRINT "3A"
2860 GOTO 60
2870 PRINT "3B"
```

```
 2880 GOTO 60
 2890 PRINT "8E"
 2900 GOTO 60
 2910 PRINT "8D"
 2920 GOTO 60
 2925 PRINT "9"
 2927 GOTO 60
 2930 PRINT "CAS INDETERMINE DE LA GRAMMAIRE , TABLE = B , E= ";E;" I= ";I
 2940 GOTO 60
 2950 END
*
```

```
"J'AI RESTE A LA MAISON"
DETERMINATION DU NIVEAU DE LA PHRASE SUIVANTE
LA PHRASE SUIVANTE EST-ELLE UNE TENTATIVE DE PROLONGATION
DE LA PHRASE PRECEDENTE?OUI-NON
?non
LA PHRASE EST-ELLE RIGIDE-VIVANTE?
?vivante
PRELEVER LE 1ER GROUPE SUJET DE LA PHRASE
DECOUPEZ-LE EN UNITES QUI SONT LES MOTS ET GROUPE
DE MOTS SUIVANTS : COMPLEMENT DETERMINATIF - COMPLEMENT
APPOSE - PRESENTATIF - GERONDIF ANTEPOSE
INTRODUISEZ LE NOMBRE D'UNITES
?2
INTRODUISEZ LES UNITES EN SEQUENCE CLOTUREES PAR *
?pp
?"*"
NIVEAU DU GROUPE SUJET  1   = 2B

Y-A-T-IL ENCORE UN GROUPE SUJET A ANALYSER?OUI-NON
?non
LA PHRASE EST-ELLE : ENONCIATIVE - NEGATIVE - IMPERATIVE
 - PASSIVE - EMPHASEE - INTERROGATIVE ?
?enonciative
LA PHRASE EST-ELLE : SIMPLE(UNE PROPOSITION PRINCIPALE)
 OU MULTIPLE(PLUSIEURS) ?
?simple
LA PHRASE COMPORTE-ELLE DES PROPOSITIONS
 SUBORDONNEES ? OUI-NON
?non
LA PHRASE COMPORTE-ELLE PLUSIEURS COMPLEMENTS ? OUI-NON
?non
DECOUPEZ LA PHRASE EN UNITES QUI SONT LES MOTS ETGROUPES
 DE MOTS SUIVANTS : COMPLEMENT DETERMINATIF - ATTRIBUT DU
 SUJET - AUXILIAIRE DU VERBE -ADVERBE
INTRODUISEZ LE NOMBRE D'UNITES
?6
INTRODUISEZ LES UNITES EN SEQUENCE - S'IL SE PRESENTE UN
 COMPLEMENT EN TETE, FAITES SUIVRE SES UNITES PAR LES
CARACTERES ST OU NST SELON QU'IL EST STEREOTYPE OU NON
 - CLOTUREZ LA SEQUENCE PAR *
?aux
?pp
?prep
?det
?nc
?"*"
```

```
NIVEAU DU GROUPE PREDICAT = 2B
"LE GENERAL ETAIT VENU POUR INSPECTER..."
DETERMINATION DU NIVEAU DE LA PHRASE SUIVANTE
LA PHRASE SUIVANTE EST-ELLE UNE TENTATIVE DE PROLONGATION
DE LA PHRASE PRECEDENTE?OUI-NON
?non
LA PHRASE EST-ELLE RIGIDE-VIVANTE?
?vivante
PRELEVER LE 1ER GROUPE SUJET DE LA PHRASE
DECOUPEZ-LE EN UNITES QUI SONT LES MOTS ET GROUPE
DE MOTS SUIVANTS : COMPLEMENT DETERMINATIF - COMPLEMENT
APPOSE - PRESENTATIF - GERONDIF ANTEPOSE
INTRODUISEZ LE NOMBRE D'UNITES
?3
INTRODUISEZ LES UNITES EN SEQUENCE CLOTUREES PAR *
?det
?nc
?"*"
NIVEAU DU GROUPE SUJET  1   = 33

Y-A-T-IL ENCORE UN GROUPE SUJET A ANALYSER?OUI-NON
?non
LA PHRASE EST-ELLE : ENONCIATIVE - NEGATIVE - IMPERATIVE
 - PASSIVE - EMPHASEE - INTERROGATIVE ?
?enonciative
LA PHRASE EST-ELLE : SIMPLE(UNE PROPOSITION PRINCIPALE)
 OU MULTIPLE(PLUSIEURS) ?
?simple
LA PHRASE COMPORTE-ELLE DES PROPOSITIONS
 SUBORDONNEES ? OUI-NON
?non
LA PHRASE COMPORTE-ELLE PLUSIEURS COMPLEMENTS ? OUI-NON
?non
DECOUPEZ LA PHRASE EN UNITES QUI SONT LES MOTS ETGROUPES
 DE MOTS SUIVANTS : COMPLEMENT DETERMINATIF - ATTRIBUT DU
 SUJET - AUXILIAIRE DU VERBE -ADVERBE
INTRODUISEZ LE NOMBRE D'UNITES
?5
INTRODUISEZ LES UNITES EN SEQUENCE - S'IL SE PRESENTE UN
 COMPLEMENT EN TETE, FAITES SUIVRE SES UNITES PAR LES
CARACTERES ST OU NST SELON QU'IL EST STEREOTYPE OU NON
 - CLOTUREZ LA SEQUENCE PAR *
?aux
?pp
?prep
?inf
?"*"
NIVEAU DU GROUPE PREDICAT = 2C
"DES SOLDATS"
DETERMINATION DU NIVEAU DE LA PHRASE SUIVANTE
LA PHRASE SUIVANTE EST-ELLE UNE TENTATIVE DE PROLONGATION
DE LA PHRASE PRECEDENTE?OUI-NON
?oui
REVET-ELLE LA FORME D-UN COMPLEMENT, D'UNE PROPOSITION
SUBORDONNEE OU D' NE PROPOSITION PRINCIPALE? OUI-NON
?non
DECOUPER LA PHRASE SELON LES MEMES UNITES QUE LA
 PHRASE PRECEDENTE
INTRODUISEZ LE NOMBRE D'UNITES
?3
```

```
INTRODUISEZ LES UNITES EN SEQUENCE, CLOTUREE PAR *
?det
?nc
?"*"
NIVEAU DU GROUPE PREDICAT = 3C
"CHEZ MONSIEUR MATEN, ON A PONCE"
DETERMINATION DU NIVEAU DE LA PHRASE SUIVANTE
LA PHRASE SUIVANTE EST-ELLE UNE TENTATIVE DE PROLONGATION
DE LA PHRASE PRECEDENTE?OUI-NON
?non
LA PHRASE EST-ELLE RIGIDE-VIVANTE?
?vivante
PRELEVER LE 1ER GROUPE SUJET DE LA PHRASE
DECOUPEZ-LE EN UNITES QUI SONT LES MOTS ET GROUPE
DE MOTS SUIVANTS : COMPLEMENT DETERMINATIF - COMPLEMENT
APPOSE - PRESENTATIF - GERONDIF ANTEPOSE
INTRODUISEZ LE NOMBRE D'UNITES
?2
INTRODUISEZ LES UNITES EN SEQUENCE CLOTUREES PAR *
?pp
?"*"
NIVEAU DU GROUPE SUJET  1   = 2B

Y-A-T-IL ENCORE UN GROUPE SUJET A ANALYSER?OUI-NON
?non
LA PHRASE EST-ELLE : ENONCIATIVE - NEGATIVE - IMPERATIVE
 - PASSIVE - EMPHASEE - INTERROGATIVE ?
?enonciative
LA PHRASE EST-ELLE : SIMPLE(UNE PROPOSITION PRINCIPALE)
 OU MULTIPLE(PLUSIEURS) ?
?simple
LA PHRASE COMPORTE-ELLE DES PROPOSITIONS
 SUBORDONNEES ? OUI-NON
?non
LA PHRASE COMPORTE-ELLE PLUSIEURS COMPLEMENTS ? OUI-NON
?non
DECOUPEZ LA PHRASE-EN UNITES QUI SONT LES MOTS ETGROUPES
 DE MOTS SUIVANTS : COMPLEMENT DETERMINATIF - ATTRIBUT DU
 SUJET - AUXILIAIRE DU VERBE -ADVERBE
INTRODUISEZ LE NOMBRE D'UNITES
?6
INTRODUISEZ LES UNITES EN SEQUENCE - S'IL SE PRESENTE UN
 COMPLEMENT EN TETE, FAITES SUIVRE SES UNITES PAR LES
CARACTERES ST OU NST SELON QU'IL EST STEREOTYPE OU NON
 - CLOTUREZ LA SEQUENCE PAR *
?prep
?np
?st
?aux
?pp
?"*"
NIVEAU DU GROUPE PREDICAT = 2E
"ALORS LE CHEF ARRIVE"
DETERMINATION DU NIVEAU DE LA PHRASE SUIVANTE
LA PHRASE SUIVANTE EST-ELLE UNE TENTATIVE DE PROLONGATION
DE LA PHRASE PRECEDENTE?OUI-NON
?non
LA PHRASE EST-ELLE RIGIDE-VIVANTE?
?vivante
PRELEVER LE 1ER GROUPE SUJET DE LA PHRASE
DECOUPEZ-LE EN UNITES QUI SONT LES MOTS ET GROUPE
DE MOTS SUIVANTS : COMPLEMENT DETERMINATIF - COMPLEMENT
APPOSE - PRESENTATIF - GERONDIF ANTEPOSE
INTRODUISEZ LE NOMBRE D'UNITES
?3
```

```
INTRODUISEZ LES UNITES EN SEQUENCE CLOTUREES PAR *
?det
?nc
?"*"
NIVEAU DU GROUPE SUJET  1    = 33

Y-A-T-IL ENCORE UN GROUPE SUJET A ANALYSER?OUI-NON
?non
LA PHRASE EST-ELLE : ENONCIATIVE - NEGATIVE - IMPERATIVE
 - PASSIVE - EMPHASEE - INTERROGATIVE ?
?enonciative
LA PHRASE EST-ELLE : SIMPLE(UNE PROPOSITION PRINCIPALE)
 OU MULTIPLE(PLUSIEURS) ?
?simple
LA PHRASE COMPORTE-ELLE DES PROPOSITIONS
 SUBORDONNEES ? OUI-NON
?non
LA PHRASE COMPORTE-ELLE PLUSIEURS COMPLEMENTS ? OUI-NON
?non
DECOUPEZ LA PHRASE EN UNITES QUI SONT LES MOTS ETGROUPES
 DE MOTS SUIVANTS : COMPLEMENT DETERMINATIF - ATTRIBUT DU
 SUJET - AUXILIAIRE DU VERBE -ADVERBE
INTRODUISEZ LE NOMBRE D'UNITES
?3
INTRODUISEZ LES UNITES EN SEQUENCE - S'IL SE PRESENTE UN
 COMPLEMENT EN TETE, FAITES SUIVRE SES UNITES PAR LES
CARACTERES ST OU NST SELON QU'IL EST STEREOTYPE OU NON
 - CLOTUREZ LA SEQUENCE PAR *
?adv
?v
?"*"
NIVEAU DU GROUPE PREDICAT = 2E
→ "C'EST  UN BEAU CHIEN"
DETERMINATION DU NIVEAU DE LA PHRASE SUIVANTE
LA PHRASE SUIVANTE EST-ELLE UNE TENTATIVE DE PROLONGATION
DE LA PHRASE PRECEDENTE?OUI-NON
?non
LA PHRASE EST-ELLE RIGIDE-VIVANTE?
?rigide
DECOUPER LA PHRASE EN UNITES QUI SONT LES MOTS
 ET GROUPES DE MOTS SUIVANTS : PRESENTATIF -
FORME IMPERSONNELLE - COMPLEMENT DETERMINATIF
INTRODUISEZ LE NOMBRE D'UNITES
?5
INTRODUISEZ LES UNITES EN SEQUENCE CLOTUREES PAR *
?pres
?ai
?adj
?nc
?"*"
 LE NIVEAU EST -1D4
→ " J'AI REGARDE UN BEAU FILM ..."
DETERMINATION DU NIVEAU DE LA PHRASE SUIVANTE
LA PHRASE SUIVANTE EST-ELLE UNE TENTATIVE DE PROLONGATION
DE LA PHRASE PRECEDENTE?OUI-NON
?non
LA PHRASE EST-ELLE RIGIDE-VIVANTE?
?vivante
PRELEVER LE 1ER GROUPE SUJET DE LA PHRASE
DECOUPEZ-LE EN UNITES QUI SONT LES MOTS ET GROUPE
DE MOTS SUIVANTS : COMPLEMENT DETERMINATIF - COMPLEMENT
APPOSE - PRESENTATIF - GERONDIF ANTEPOSE
INTRODUISEZ LE NOMBRE D'UNITES
?2
```

```
INTRODUISEZ LES UNITES EN SEQUENCE CLOTUREES PAR *
?pp
?"*"
NIVEAU DU GROUPE SUJET  1   = 2B

Y-A-T-IL ENCORE UN GROUPE SUJET A ANALYSER?OUI-NON
?non
LA PHRASE EST-ELLE : ENONCIATIVE - NEGATIVE - IMPERATIVE
 - PASSIVE - EMPHASEE - INTERROGATIVE ?
?enonciative
LA PHRASE EST-ELLE : SIMPLE(UNE PROPOSITION PRINCIPALE)
 OU MULTIPLE(PLUSIEURS) ?
?simple
LA PHRASE COMPORTE-ELLE DES PROPOSITIONS
 SUBORDONNEES ? OUI-NON
?non
LA PHRASE COMPORTE-ELLE PLUSIEURS COMPLEMENTS ? OUI-NON
?non
DECOUPEZ LA PHRASE EN UNITES QUI SONT LES MOTS ETGROUPES
 DE MOTS SUIVANTS : COMPLEMENT DETERMINATIF - ATTRIBUT DU
 SUJET - AUXILIAIRE DU VERBE -ADVERBE
INTRODUISEZ LE NOMBRE D'UNITES
?6
INTRODUISEZ LES UNITES EN SEQUENCE - S'IL SE PRESENTE UN
 COMPLEMENT EN TETE, FAITES SUIVRE SES UNITES PAR LES
CARACTERES ST OU NST SELON QU'IL EST STEREOTYPE OU NON
 - CLOTUREZ LA SEQUENCE PAR *
?aux
?pp
?det
?adj
?nc
?"*"
NIVEAU DU GROUPE PREDICAT = 3A
"AVEC DES SOLDATS "
DETERMINATION DU NIVEAU DE LA PHRASE SUIVANTE
LA PHRASE SUIVANTE EST-ELLE UNE TENTATIVE DE PROLONGATION
DE LA PHRASE PRECEDENTE?OUI-NON
?oui
REVET-ELLE LA FORME D-UN COMPLEMENT, D'UNE PROPOSITION
SUBORDONNEE OU D'UNE PROPOSITION PRINCIPALE? OUI-NON
?oui
REGROUPER LA PHRASE ORALE PRECEDENTE ET LA PHRASE
 PRESENTE EN UNE
Y-A-T-IL DE NOUVEAUX GROUPES SUJET ? OUI-NON
?non
LA PHRASE EST-ELLE : ENONCIATIVE - NEGATIVE - IMPERATIVE
 - PASSIVE - EMPHASEE - INTERROGATIVE ?
?enonciative
LA PHRASE EST-ELLE : SIMPLE(UNE PROPOSITION PRINCIPALE)
 OU MULTIPLE(PLUSIEURS) ?
?simple
LA PHRASE COMPORTE-ELLE DES PROPOSITIONS
 SUBORDONNEES ? OUI-NON
?non
LA PHRASE COMPORTE-ELLE PLUSIEURS COMPLEMENTS ? OUI-NON
?oui
DECOUPEZ LA PHRASE EN UNITES QUI SONT LES COMPLEMENTS -
CONJONCTION - VERBE
INTRODUISEZ LE NOMBRE D'UNITES
?4
```

```
INTRODUISEZ LES UNITES EN SEQUENCE CLOTUREES PAR *
?v
?ce
?cs
?"*"
NIVEAU DU GROUPE PREDICAT = 5B
" GENTIL CHAT "
DETERMINATION DU NIVEAU DE LA PHRASE SUIVANTE
LA PHRASE SUIVANTE EST-ELLE UNE TENTATIVE DE PROLONGATION
DE LA PHRASE PRECEDENTE?OUI-NON
?non
LA PHRASE EST-ELLE RIGIDE-VIVANTE?
?rigide
DECOUPER LA PHRASE EN UNITES QUI SONT LES MOTS
 ET GROUPES DE MOTS SUIVANTS : PRESENTATIF -
FORME IMPERSONNELLE - COMPLEMENT DETERMINATIF
INTRODUISEZ LE NOMBRE D'UNITES
?3
INTRODUISEZ LES UNITES EN SEQUENCE CLOTUREES PAR *
?adj
?nc
?"*"
LE CAS N-EST PAS DEFINI:TABLE = T,E =  1  I =  12
"ALORS ILS S'ETAIENT BATTUS, ALORS ILS ONT ETE DANS LA MAISON"
DETERMINATION DU NIVEAU DE LA PHRASE SUIVANTE
LA PHRASE SUIVANTE EST-ELLE UNE TENTATIVE DE PROLONGATION
DE LA PHRASE PRECEDENTE?OUI-NON
?non
LA PHRASE EST-ELLE RIGIDE-VIVANTE?
?vivante
PRELEVER LE 1ER GROUPE SUJET DE LA PHRASE
DECOUPEZ-LE EN UNITES QUI SONT LES MOTS ET GROUPE
DE MOTS SUIVANTS : COMPLEMENT DETERMINATIF - COMPLEMENT
APPOSE - PRESENTATIF - GERONDIF ANTEPOSE
INTRODUISEZ LE NOMBRE D' NITES
?2
INTRODUISEZ LES UNITES EN SEQUENCE CLOTUREES PAR *
?pp
?"*"
NIVEAU DU GROUPE SUJET  1   = 2B

Y-A-T-IL ENCORE UN GROUPE SUJET A ANALYSER?OUI-NON
?oui
DECOUPEZ-LE EN UNITES QUI SONT LES MOTS ET GROUPE
DE MOTS SUIVANTS : COMPLEMENT DETERMINATIF - COMPLEMENT
APPOSE - PRESENTATIF - GERONDIF ANTEPOSE
INTRODUISEZ LE NOMBRE D' NITES
?2
INTRODUISEZ LES UNITES EN SEQUENCE CLOTUREES PAR *
?pp
?"*"
NIVEAU DU GROUPE SUJET  2   = 2B

Y-A-T-IL ENCORE UN GROUPE SUJET A ANALYSER?OUI-NON
?non
LA PHRASE EST-ELLE : ENONCIATIVE - NEGATIVE - IMPERATIVE
 - PASSIVE - EMPHASEE - INTERROGATIVE ?
?enonciative
LA PHRASE EST-ELLE : SIMPLE(UNE PROPOSITION PRINCIPALE)
 OU MULTIPLE(PLUSIEURS) ?
?multiple
DECOUPEZ LA PHRASE EN UNITES QUI SONT LES PHRASES ET CONJONCTION
INTRODUISEZ LE NOMBRE D'UNITES
?4
```

```
INTRODUISEZ LES UNITES EN SEQUENCE CLOTUREES PAR *
?p
?conj
?p
?"*"
NIVEAU DU GROUPE PREDICAT = 78
```

Conclusion.

Nous voyons avec quelle facilité nous avons pu programmer cette application, grâce à l'utilisation des tables séquentielles. Nous n'avons dû établir, pour cette application, qu'un petit organigramme pour la consultation.

4. Annexe : Echelle de structures pour une évaluation du langage oral.

PARTIE I.

Structures rigides ou semi-rigides. (le groupe sujet ne se distingue pas du groupe prédicat).

NIVEAU-I : a. mot phrase : ex. chien, pomme

b. SN (syntagme nominal = article + nom(N)) :

1. article indéfini : ex. un chien
2. article défini : ex. le chien
3. article partitif : ex. du pain

c. présentatif :

1. + N propre : voilà Pierre
2. + SN (article + nom commun) : c'est un chien
3. + SN (déterminant + nom commun) : c'est mon chien
4. + SN (déterminant + N + expansion : adjectif, complément déterminatif) c'est la maison de Pierre.

d. Tours impersonnels : il pleut.

PARTIE II.

Structures de phrases vivantes. (groupe sujet + groupe prédicat)

Groupe du sujet.

NIVEAU 0 : Pas de sujet.

NIVEAU I : Sujet = SN (nom propre locuteur)

NIVEAU II : a. SN (nom propre $\neq$ locuteur).

1. incorporé dans un tour présentatif :
   c'est Pierre qui ....
2. repris par un pronom relais :
   Pierre, il ....
3. non repris, ni incorporé.

b. S pronominal : je, tu, il, on ...

NIVEAU III : SN(déterminant (article, possessif, démonstratif) + N commun).

1. incorporé dans un tour présentatif :
   c'est le chien ....
2. repris par un pronom relais :
   le chien, il ...
3. ni repris, ni incorporé.

NIVEAU IV : SN + expansion.

a. déterminant + adjectif + N : la petite fille

b. déterminant + N + complément déterminatif :
   le chat du voisin

c. déterminant + post-article + N : le même homme

   ou

   pré-article + déterminant + N : tous les gens

d. déterminant + N + SN apposé : un garçon,
   mon ami

e. structures précédentes combinées entre elles :
   la petite fille du voisin.

   Pour chacune de ces structures :

   1. incorporé dans tous présentatif
   2. repris par pronom relais
   3. ni repris, ni incorporé.

f. plusieurs sujets simples

h. plusieurs sujets expansés.

NIVEAU V : SN + expansion par une proposition

a. relative introduite par QUI : l'homme qui arrive

b. relative introduite par QUE : l'homme que je vois

c. relative introduite par d'autres relatifs.

NIVEAU VI : Mise en relief d'un élément du groupe sujet

a. gérondif antéposé : en chantant, je ....
b. participe antéposé : arrivés, ils ....
c. tour démonstratif : c'est lui qui ....
d. extraction du sujet : ce type-là, c'est un drôle.

Groupe du prédicat : (énonciation, négation, interrogation à forme énonciative, impératif).

NIVEAU I : SV (syntagme verbal) = V (verbe) : je danse
ou
(auxiliaire + infinitif) : j'ai été voir

NIVEAU II : SV = V + SN

a. SN = COD, CC sans préposition, attribut
b. SN = syntagme prépositionnel + COI, CC complément de verbe : je frappe à la porte
c. SN = syntagme prépositionnel + infinitif: il vient pour jouer
d. SN = syntagme pronominal devant le verbe : je le vois
e. SN = CC de phrase mobile mais stéréotypé dans son emploi : le soir, je rentre
ou
CC sous forme d'adverbe : je mange vite.

NIVEAU III : SV = V + SN + expansion

a. SN = déterminant + adjectif + N (+ adjectif)
b. SN = déterminant + N + complément déterminatif
c. SN = syntagme prépositionnel + infinitif + SN (CO, CC) : nous venons pour voir la maison
d. SN = déterminant + N + attribut du complément : j'appelle mon chat Poussy.

NIVEAU IV : SV = V + SN + SN (voir la nature des SN ci-dessus)

a. compléments différents hiérarchisés
b. compléments coordonnés
c. double pronomalisation.

NIVEAU V : SV = V + SN [+ expansion] + SN [+ expansion]
(une seule des expansions)

a. second complément expansé : avec son père et son petit frère

b. premier complément expansé : il a frappé le chat noir avec un bâton.

NIVEAU VI : SV = V + SN + expansion + SN + expansion.

NIVEAU VII : deux structures des types ci-dessus accolées sans interruption :

a. juxtaposées

b. coordonnées.

NIVEAU VIII : une structure d'un des types précédents avec expansion = proposition subordonnée.

a. relative introduite par QUI

b. relative introduite par QUE

c. conjonctive introduite par QUE

d. infinitive avec SN sujet : je vois le chien courir

e. infinitive avec syntagme pronominal antéposé sujet : je le vois courir

f. interrogative indirecte : je sais comment on fait

g. circonstantielle.

NIVEAU IX : phrase d'un des types précédents avec CC de phrase mobile non stéréotypé : avec mon frère et ma soeur nous avons joué.

NIVEAU X :

a. interrogation avec inversion

c. emphase

d. phrase passive.

APPLICATION V : Procédure de gestion des échanges en télétraitement par caractères de contrôle.

Le problème de la transmission correcte des informations entre ordinateurs et/ou terminaux nécessite l'introduction de caractères spéciaux, dans le message et dans la procédure. Ils sont destinés à assurer une bonne compréhension du dialogue qui va s'établir entre les deux stations.

Nous considérerons le cas de la transmission en mode de contention. Ce mode de transmission ne peut s'opérer qu'en point à point : lorsqu'un des deux ordinateurs/terminaux désire obtenir la ligne pour envoyer un message à l'autre, il envoie le caractère de contrôle 'ENQ' et prend le statut de "maître". L'ordinateur/terminal récepteur prend le statut d'"esclave".

Les caractères de contrôle principaux sont les suivants :

STX	précède tout bloc de message
ETX	termine tout message
ETB	termine tout bloc de message
EOT	fin de transmission
ENQ	demande l'état d'une station; la répétition de transmission. (En cas de conflit entre des ENQ transmis simultanément, un test de priorité déterminera quelle station a le droit de transmettre).
ACK0	accusé de réception positif pour les blocs pairs
ACK1	accusé de réception positif pour les blocs impairs
NAK	accusé de réception négatif
WABT	demande d'un délai de la part du récepteur
TTD	demande d'un délai de la part de l'émetteur.

Parmi les séquences que peuvent s'échanger les deux stations, nous trouvons un certain nombre de séquences types; relevons-en quelques unes.

1. Envoi d'un message de la station A (maître) vers la station B (esclave). (le message est envoyé par blocs).

Phases	STATION A		STATION B	Commentaires
1	ENQ	→		A demande à B s'il est prêt à recevoir
2				
		←	ACK0	B répond qu'il est prêt
3				
27	ETB bloc n°1 STX	→		Transmission du bloc n° 1
28				
		←	ACK1	Réponse "bien reçu"
7	ETB bloc n°2 STX	→		Transmission du bloc n° 2
8				
		←	NAK	Réponse "mal reçu" (erreur de parité longitudinale)
15				
16	ETB bloc n°2 STX	→		Retransmission du bloc n° 2
8				
		←	ACK0	Réponse "bien reçu"
27	ETX bloc n°3 STX	→		Transmission du 3ème et dernier bloc
29				
		←	ACK1	Réponse "bien reçu"
4	EOT	→		Fin de transmission et déconnection

Remarque :

En cas de mauvais accusé de réception, un bloc peut être retransmis jusqu'à 3 fois; au-delà, un caractère EOT est envoyé par l'émetteur et la ligne déconnectée.

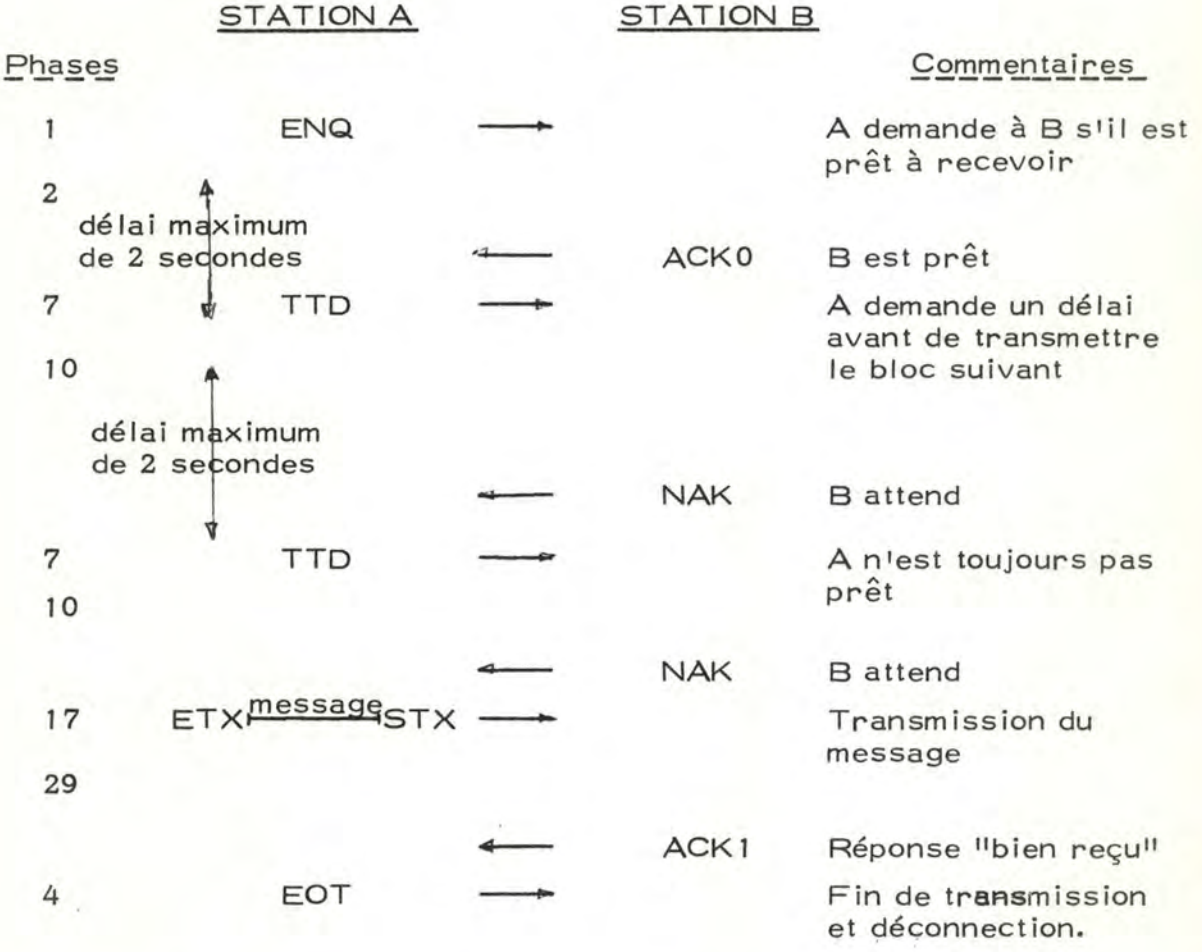
2. Le récepteur ne répond pas.

Phases	STATION A.		STATION B	Commentaires
1	ENQ	→		A demande à B s'il
2				est prêt à recevoir
		↕ délai de 3 secondes (= temporisation T_1)		B ne répond pas
6				
1	ENQ	→		idem
2				
6				
1	ENQ	→		idem
2				
6				
4	EOT	→		Fin de transmission et déconnection après 3 essais

3. Le récepteur demande un délai.

1	ENQ	→		A demande à B s'il
2				est prêt à recevoir
		←	ACK0	B est prêt
3				
27	ETB bloc n°1 STX	→		Transmission du bloc n° 1
28				
			↕ délai maximum de 2 secondes	
37		←	WABT	B demande un délai avant réception du bloc suivant
39	ENQ	→		A demand l'état de B
28				
		←	ACK1	B est prêt à recevoir et signale que le dernier bloc a été bien reçu
7	ETX bloc n° 2 STX	→		Transmission du 2ème et
9				dernier bloc
		←	ACK0	Réponse "bien reçu"
4	EOT	→		Fin de transmission et déconnection.

4. L'émetteur demande un délai.



Nous allons mettre en page cette procédure d'échange entre deux stations dans une table de décision séquentielle. En réalité, nous devons en établir deux : une qui rend compte de ce qu'émet et reçoit le maître et une autre de ce qu'émet et reçoit l'esclave. Nous n'établirons que la première, la deuxième pratiquement s'en déduit, étant son complément.

Les séquences qui seront enregistrées dans ces 2 tables sont caractérisées par le fait qu'elles contiennent chacune en alternance une opération d'écriture (caractère de contrôle et/ou message) suivie d'une opération de lecture (accusé de réception).

Table M (table du maître).

Nous commençons par y enregistrer les séquences définies ci-dessus. Ensuite, examinons les cases libres : elles peuvent être l'indice de phases omises parce qu'elles ne se produisent jamais mais,

moyennant une analyse plus poussée, nous pouvons découvrir certaines séquences ou sous-séquences nouvelles. En tous cas, il est souhaitable de combler certaines cases vides susceptibles d'être sollicitées accidentellement. Elles le seront au moyen d'un symbole qui conduira le système vers un état d'alarme.

Montrons comment les séquences sont enregistrées dans la table en parcourant une séquence particulière : la phase initiale provoque l'envoi du caractère de contrôle 'ENQ' avec basculement en phase 2 de façon à réceptionner normalement un ACKO. S'il s'agit bien de ce caractère, le premier bloc peut être transmis (phase 27) mais dans des conditions de contrôle différentes selon que le test indique qu'il se termine par ETB ou ETX. Cette opération d'écriture sera suivie d'une lecture de caractère (en phases 28, 29, 30); selon le caractère reçu la poursuite de la séquence sera différente. Ainsi, si un ETB/ETX a été transmis et qu'un caractère invalide a été reçu, un test du nombre d'ENQ émis aura lieu en phases 31, 32. Si ce nombre est inférieur à 3, une nouvelle interrogation par ENQ (en phases 33, 34) aura pour but de demander à la station distante, d'accuser réception. Et ainsi de suite.

N'oublions pas de remarquer que cette table contient les phases équivalentes suivantes :

11 = 27,	12 = 28,	13 = 29,	14 = 30
31 = 37,	32 = 38,	33 = 39,	34 = 40

Ces phases définissent un seul et même état de la procédure et permettent de réduire la table en fusionnant leur ligne.

			CONDITIONS															
				caractère contrôle émis			caractère contrôle reçu						compteur ENQ		compteur bloc		test priorité	? écoulé
	PHASES	DECISIONS	B. I.	ETB	ETX	TTD	ENQ	ACK0	ACK1	NAK	WABT	invalide	<3	>3	<3	>3	M	E	
Demande d'obtention de la ligne pour envoyer un message	1	émission de 'ENQ', initialiser et lancer T_1 Augmenter le compteur ENQ d'une unité.	2															
Attente de la réponse à la demande d'occupation de ligne	2	Basculer dans l'état réception					5	3		4		6						6
L'esclave est prêt à recevoir le message	3		27															
Fin de transmission	4	Emission de 'EOT'	FIN															
Les 2 stations demandent la ligne	5																1	41
Pas de réponse ou réponse invalide à la demande d'envoi de message	6												1	4				
Un bloc pair peut être envoyé	7	Emission d'un bloc de message, initialiser et lancer T_1 Mettre le compteur ENQ à 0, Mettre le compteur bloc à 1.		8	9	10												
Attente de l'accusé de réception pour un bloc pair clôturé par ETB	8	Basculer dans l'état réception						27	11	15	17	11						11
Attente de l'accusé de réception pour un bloc pair clôturé par ETX	9	Basculer dans l'état réception						4	12	15	18	12						12
Délai demandé par l'émetteur	10	Basculer dans l'état réception								7								
Accusé de réception invalide pour un bloc pair clôturé par ETB	11												13	4				
Accusé de réception invalide pour un bloc pair clôturé par ETX	12												14	4				
Demande de l'accusé de réception pour un bloc pair clôturé par ETB	13	Emission de 'ENQ', initialiser et lancer T_1, Augmenter le compteur ENQ de 1.	8															
Demande de l'accusé de réception pour un bloc pair clôturé par ETX	14	Emission de 'ENQ', initialiser en lancer T_1, Augmenter le compteur ENQ de 1.	9															
Accusé de réception négatif pour un bloc	15														16	4		
Le bloc pair peut être réémis	16	Réémission du dernier bloc, initialiser et lancer T_1, Augmenter le compteur bloc de 1.		8	9	10												
Délai demandé par le récepteur avant l'envoi de l'accusé de réception d'un bloc pair clôturé par ETB	17												19	4				
Délai demandé par le récepteur avant l'envoi de l'accusé de réception d'un bloc pair clôturé par ETX	18												20	4				
Demande si le récepteur est prêt à envoyer l'accusé de réception d'un bloc pair clôturé par ETB	19	Emission de 'ENQ', initialiser et lancer T_1 Augmenter le compteur ENQ de 1.	8															
Demande si le récepteur est prêt à envoyer l'accusé de réception d'un bloc pair clôturé par ETX	20	Emission de 'ENQ' initialiser et lancer T_1 Augmenter le compteur ENQ de 1.	9															
Un bloc impair peut être envoyé	27	Emission d'un bloc de message initialiser et lancer T_1, Mettre le compteur ENQ à 0, Mettre le compteur bloc à 1.		28	29	30												
Attente accusé de réception pour un bloc pair clôturé par ETB	28	Basculer dans l'état réception						31	7	35	37	31						31
Attente accusé de réception pour un bloc pair clôturé par ETX	29	Basculer dans l'état réception						32	4	35	38	32						32
Délai demandé par l'émetteur	30	Basculer dans l'état réception								27								
Accusé de réception invalide pour un bloc impair clôturé par ETB	31												33	4				
Accusé de réception invalide pour un bloc impair clôturé par ETX	32												34	4				
Demande de l'accusé de réception pour un bloc impair clôturé par ETB	33	Emission de 'ENQ', initialiser et lancer T_1, Augmenter le compteur ENQ de 1	28															
Demande de l'accusé de réception pour un bloc impair clôturé par ETX	34	Emission de 'ENQ' initialiser et lancer T_1, Augmenter le compteur ENQ de 1.	29															
Accusé de réception négatif pour un bloc impair	35														36	4		
Le bloc impair peut être réémis	36	Réémission du dernier bloc. initialiser et lancer T_1 Augmenter le compteur bloc de 1.		28	29	30												
Délai demandé par le récepteur avant l'envoi de l'accusé de réception d'un bloc impair clôturé par ETB	37												39	4				
Délai demandé par le récepteur avant l'envoi de l'accusé de réception d'un bloc impair clôturé par ETX	38												40	4				
Demande de l'accusé de réception d'un bloc impair clôturé par ETB	39	Emission de 'ENQ' initialiser et lancer T_1 Augmenter le compteur ENQ de 1.	28															
Demande de l'accusé de réception d'un bloc impair clôturé par ETX	40	Emission de 'ENQ' initialiser et lancer T_1 Augmenter le compteur de 1	29															
L'esclave devient le maître	41		1															

CONCLUSION.

Nous avons présenté la table de décision séquentielle et nous l'avons étudiée en tant qu'outil de synthèse des procédures séquentielles. Au terme de ce travail, il nous paraît bon d'émettre quelques critiques la concernant. Initialement, elle nous était inconnue, à présent jugeons-la :

- la table, outil de mise en page :

  nous avions annoncé que nous cherchions un outil de mise en page plus manipulable et plus concis que l'organigramme. La table de décision séquentielle satisfait-elle ces deux points ?

  D'une part, de par sa forme tabulaire, elle est plus manipulable que l'organigramme.

  D'autre part, elle peut revêtir deux formes : détaillée ou condensée. La forme détaillée de la table prend de l'ampleur aussi rapidement que l'organigramme; elle n'est qu'une image de ce dernier exprimé en terme de phase. De plus, quel que soit le nombre de conditions qui articulent la procédure, chaque ligne n'occupe que deux colonnes; ceci entraîne une perte de place considérable. Si la procédure s'y prête (cas par exemple, des procédures dont les traitements à exécuter sont fonction de symboles ou caractères lus ou apparaissant successivement), il y a avantage à l'enregistrer dans une table condensée. Celle-ci est toujours plus concise que l'organigramme; pensez à la mise en page sous forme d'organigramme de la procédure de gestion des échanges au télétraitement. La table, dans sa forme condensée, permet de présenter de manière claire et concise certaines procédures qui nécessitent plusieurs pages de définition ou plusieurs pages d'organigramme.
  La justification de la table en tant qu'outil de mise en page est donc sa forme condensée.
  Elle possède également l'avantage d'être lisible et compréhensible par tous, informaticiens et non informaticiens.

- la table, outil d'analyse :

  grâce à la théorie des graphes et au graphe qu'il est toujours possible de définir à partir d'une table, il est aisé de développer toutes les séquences et branches de la procédure enregistrée dans cette table. La table d'elle-même fait également ressortir tous les cas omis. Par contre, il est malaisé de vérifier si dans un organigramme toutes les possibilités ont été envisagées ou de déterminer tous les chemins possibles de la procédure.
  Si, après analyse, des modifications sont à apporter à la procédure et/ou à la table, il n'y aura aucune difficulté à introduire dans la table de nouvelles phases et de les raccrocher aux phases déjà définies, ou d'en supprimer.

- la table, outil de simplification :

  une procédure enregistrée, dans une table de décision séquentielle, peut donner naissance à des phases équivalentes. Celles-ci détectées peuvent être fusionnées afin de simplifier la procédure et réduire la table.

- la table, outil d'aide à la programmation :

  la table de décision séquentielle permet non seulement de présenter au programmeur une synthèse claire et précise du problème, elle lui permet également de programmer d'une manière soignée et avantageuse pour l'exploitation, la maintenance et les corrections d'erreur.

Il est normal de comparer la table de décision séquentielle à l'organigramme, seul outil connu pour la synthèse des procédures séquentielles en logique enregistrée.

Notre but n'est pas de détrôner l'organigramme mais de montrer que :

- dans certains cas, il est plus avantageux de mettre en page les procédures séquentielles dans une table séquentielle;
- la table séquentielle est un meilleur outil d'analyse et de simplification que l'organigramme;
- elle permet de procéder à une programmation soignée, modulaire et classique.

La notion fondamentale qui a permis de développer l'idée de table de décision séquentielle, outil de synthèse, est la notion de phase.

BIBLIOGRAPHIE

CHAPITRE I :

- System analysis techniques : "Decision tables, what, why and how"

  COUGER and KNAPP
  J. Wiley 1974

- Les tables de décision.

  G. BAGLIN et J. KLEE
  Entreprise moderne d'édition 1970.

- "Les tables de décision".

  J. L. PIQUART
  Revue Informatique et gestion n° 3-4-5.

- L'organigramme de Programmation.

  CENTI
  Centre de traitement de l'information 1974.

CHAPITRE II :

- Logique combinatoire et séquentielle.

  J. LAGASSE
  Dunod 1971.

- Logique binaire et commutation.

  J. BRUNIN
  Dunod 1966.

- Systèmes logiques (Tome II)

  PERRIN - DENOUETTE - DACLIN
  Dunod 1967.

CHAPITRE III :

- Théorie des graphes.

  J. FICHEFET
  Notes de cours, FNDP Namur.

CHAPITRE V :

- Compiler construction for digital computers.

  D. GRIES
  J. Wiley 1971.

- The anatomy of a compiler.

  J. LEE
  Reinhold book corporation 1967.

- Translation of computer languages.
WEINGARTEN
Holden-day, Inc. 1963.

- Théorie des grammaires formelles.
H. LEROY
Notes de cours, FNDP Namur.

- Introduction au télétraitement.
J. P. WINDAL
Notes de cours, FNDP Namur.